Modeling the Effects of Intelligent Driver Assistance on Driver Behaviors

by

Ganesh Raghunath Deo

A thesis submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Master of Science

Auburn, Alabama
August 9, 2010

Keywords: Cognitive Workload, Driver Modeling, Simulation Experiment, Telematics
Copyright 2010 by Ganesh Raghunath Deo

Approved by

N. Hari Narayanan, Chair, Professor of Computer Science and Software Engineering
David Umphress, Associate Professor of Computer Science and Software Engineering
Kai Chang, Professor and Chair of Computer Science and Software Engineering

Abstract

Many studies have concluded that activities like answering a cell phone call and entering an address while driving could potentially distract the driver. This thesis presents the design and experimental evaluation by simulation of four Intelligent Driver Assistance (IDA) modes to improve driver safety and efficiency. In order to verify the effectiveness of these modes, we conducted four experiments on a driver simulator called the Cognitive-Computational Driving Model (CCDM). The four IDA modes are: 1) Limited Cognitive Resources: None of the driver's workload channels (such as vision, speech, cognition etc.) are allowed to exceed the baseline values needed for normal driving. 2) Safe Mode driving: Distracting events such as cell phone calls are terminated, if a safety-critical event like a vehicle cutting in front occurs. 3) Dynamic delay: Distracting events (such as entering GPS destination by voice) are delayed or disabled temporarily, if a safety-critical event (such as an obstacle in lane ahead) occurs. 4) Static delay: Distracting events (e.g., cell phone rings) are delayed by a certain time interval if the driver's workload is high.

We measured various aspects of the driver such as cognitive load, time to complete safety-critical goals, ratio of interface related actions to driving related actions, reaction times, visual perception delays, etc. with and without intelligent driver assistance. After analyzing the data, we found that in the simulated scenarios, with intelligent driver assistance the driver could concentrate more on safety-critical tasks and finished them earlier compared to when intelligent driver assistance was turned off. Cognitive overload was reduced in all modes except the dynamic delay mode. Visual overload was less with limited cognitive resources and static delay modes. Also, we found that among the four modes of IDA, the static delay mode was better than the rest of the modes in terms of safety, efficiency, and visual and cognitive resource usage of the driver.

We integrated the kernel of simulation engine with Microsoft Excel 2007. This allowed us to pipe the simulation output data at various intervals to the Excel application. With the office-automation feature of CLI/C++, we have been able to automatically generate driver's workload charts/graphs from each simulation run.

Acknowledgments

I am thankful to everyone who stood behind me and encouraged me during my study at Auburn University. My special thanks to major advisor, Dr. N. Hari Narayanan for his continuous help and support during my research. I am grateful to Dr. Kai Chang and Dr. David Umphress for serving on my committee. I greatly appreciate people at Electronics & Telecommunications Research Institute (ETRI), Korea for providing funding for this research. Finally, my sincere thanks to my parents; without their blessings this work wouldn't have been possible.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction and Research Questions

In today's modern world of information technology, if we need to measure a driver's behavior while driving with various distractions occurring inside or outside of the car; there are various techniques available to do this - mounting cameras at various positions inside the car, mounting an eye tracker and sensors on the drivers body, and allowing the driver to drive on virtual driving simulators with the help of computers. Though these measures can provide accurate data on human drivers, these types of methods are expensive and may require legal permission as it involves human subjects. Therefore, we developed a computational simulation of a virtual driver and his vehicle, based on cognitive and psychomotor parameters reported in the literature from experiments with human drivers in actual or simulated driving scenarios. This Cognitive-Computational Driving Model (CCDM) is customizable; it could be configured to model young, adult or old drivers driving under different weather conditions. The CCDM includes an Intelligent Driver Assistance (IDA) subsystem; this subsystem can automatically detect any activities that can cause distraction to the driver while the driver is engaged in an emergency maneuver such as avoiding a vehicle in front that suddenly stopped. It either prevents distracting events from taking place or it delays them until emergency maneuvering tasks finish. This way, the driver's safety is ensured.

Many automobiles today have some type of intelligent system installed inside them to assist the driver in dealing with various situations. If we wish to evaluate the effectiveness of these systems, our driving simulation could be a good option for various reasons: it is customizable, it is integrated with Microsoft Excel to automatically generate simulation traces and human performance graphs, and it can quickly run different scenarios and produce results. Therefore, in a matter of few days, we can model a proposed intelligent assistance technique and run experiments to see if the new intelligent assistance technique may be beneficial or not. Moreover, the CCDM does not require human participants as a part of experiments; instead, the simulation plausibly models different driver characteristics.

In the research reported in this thesis, we have proposed four research questions as follows:

1) If an IDA system is able to prevent the driver from doing actions that will overload the visual, auditory or cognitive channel, what effect will such a system have on driving safety when the driver has to deal with an emergency while operating a telematics device (or starts to operate a device while dealing with an emergency)?
2) If an IDA system is able to delay the driver from operating a telematics device while engaged in an emergency maneuver until that maneuver is completed, what effect will such a system have on driving safety?
3) If an IDA system is able to detect when the driver is engaged in an emergency maneuver and can make the driver abandon all other non-emergency driving and telematics actions, what effect will such a system have on driving safety?
4) If an IDA system is able to delay the operation of a telematics device (e.g., delay cell phone ringing) by a fixed amount of time while the driver is engaged in an emergency maneuver, what effect will such a system have on driving safety?

To answer the above research questions, we simulated four different IDA modes:

1) Cognitive resource limits.

2) Dynamic delay for safety.

3) Safe mode driving.

4) Static delay for safety.

We then evaluated each of these modes by running a simulation multiple times and measured their effectiveness with ten metrics regarding driver safety and sufficiency. The following chapters describe this research in greater detail.

Chapter 2

Literature Survey

Many studies conducted by researchers have suggested that secondary tasks such as entering GPS address and talking on a cell phone while driving can significantly affect the driver's performance. An investigation done by Dario D. Salvucci et al. [1] pointed out that even if these secondary tasks decrease driver performance, the important point is how often the driver shifts his attention from driving to other tasks and back to driving again. An experiment was conducted in which participants were given a task of dialing a cell phone while driving in a driving simulator. Average dialing time was calculated when participants were told to focus on steering (making sure the vehicle stays in center lane), and when told to quickly finish the dialing. Average lateral deviation from central lane at varying speeds was calculated when participants were instructed to focus on steering, and told to quickly finish dialing. The results showed that when the participants were told to focus on steering, dialing time was increased. When participants were instructed to focus on quickly finishing dialing, lateral deviation from central lane was increased. This study suggested that telematics devices that allow driver to engage in secondary tasks of small time intervals are better than devices which require longer interaction intervals.

In recent times, automobile manufacturers have been developing in-vehicle driver assistance systems that could assist drivers in dealing with various situations while driving. Research done by Xiaoming Zhang et al. [2] discusses two types of driver assistance systems: 1) classical driver assistance, and 2) cognitive driver assistance. The former system gets data from sensors, processes the data and then decides whether to display a message to the driver or not. The latter system consists of a cognition module and a decision module. The cognition module receives data from sensors, extracts information from the data and tries to match it with the driver's current state. The decision module selects correct driver assistance model, tries to match the model with the driver's current state, and decides if there is any need to display a message to the driver. Xiaoming Zhang et al. developed three prototype systems for driver assistance: 1) the driver has control and requests assistance from the system; 2) the assistance system initiates automatically and requests authorization from driver; and 3) the assistance system initiates automatically and starts driving.

Changxu Wu et al. [3] at the University of Michigan have developed a prototype of an Adaptive Workload Management System (AWMS) for vehicles called "Queuing Network-Model of Human Processor" (QN-MHP). This is more sophisticated than existing AWMS that do not use the driver's workload to control the frequency of messages generated from in-vehicle information systems. The prototype is composed of two parts: 1) QN-MHP and 2) Message Controller. The Message Controller interacts with QN-MHP to generate messages with delays at appropriate times based on driving conditions like speed, curves etc. To test the validity of the prototype, an experiment was conducted in which participants' main task was to read the speed of the target vehicle while there were two concurrent secondary tasks: listen to a radio announcement and press a key on the screen accordingly. Measures of perceived workload were obtained by serialization and randomization of order of secondary tasks. The results indicated that with serialization of subtasks, perceived workload was significantly less compared to randomization. Also, standard deviation of lane position was more in randomization of secondary tasks.

A Driver Assistance System (DAS) developed at Honda R&D Co., Japan [4] improves driver's safety and reduces driver's workload. This DAS mainly performs two tasks: keep the vehicle in the center of the lane and keep the speed matching the nearby vehicles. Driver's mental and physical workload was measured using electromyography (EMG) and electro dermal activity (EDA) in the presence and absence of DAS. To verify the effectiveness of DAS, a test was conducted on three male drivers; their task was to follow the front vehicle which was accelerating or decelerating at certain times during the test. It was found that steering torque was less in the presence of DAS, hence DAS assisted in safe driving. It was also found that with DAS the driver was infrequently and less tense compared to when DAS was absent.

There are various ways in which a driver can be warned about dangerous situations while driving, by providing visual information on a screen, providing an auditory beep, a vibrating seat, etc. Jennifer F. May and Carryl L. Baldwin from Old Dominion University and Raja Parasuraman from George Mason University [5] investigated the effectiveness of two auditory Collision Avoidance Systems in drivers with task-induced fatigue. Fatigue can adversely affect a driver's ability to accelerate, decelerate, and keep in a lane. There were three hypotheses that were considered. 1) Age, gender, and CAS type would significantly affect response to CAS (crash or stopped). 2) Stopping distance would be more with CAS warning. 3) Participants who crashed would have higher speed at the time of crash. In the experiment conducted to validate these hypotheses, participants of different age groups were induced to fatigue by continuous simulated driving for about 90 minutes. Then front car's speed was suddenly reduced and then three CAS conditions occurred (no warning, warning of 1000 Hz tone, and verbal warning). Results showed that overall 18% of drivers crashed when "no warning" and 11% of drivers crashed with auditory warning. So the crash rate was significantly reduced when auditory warning was used. Also, this study found that auditory warnings significantly improved safety for older drivers than drivers with age less than 35.

A research report published by the IBM Research Division [6] discusses various aspects of how driver’s safety could be increased. The report describes the following components that could be used in future vehicles. 1) Voice Controlled Interface: many studies have concluded that interaction with in-vehicle telematics devices using voice control increases drivers' safety. Researchers at IBM have been developing a “Quasi-Natural Language Understanding” speech processing unit that would understand and execute simple voice commands initiated by the driver. 2) Speech Recognition: authors have proposed some ideas that could be used to do speech recognition with low processing power CPU’s inside the vehicle. 3) Artificial Passenger: this is a new concept. A solitary driver's fatigue can be reduced by having an artificial passenger interacting with the driver by providing him mental stimulation (asking him to play games, puzzles etc.) and physical stimuli (warning or bell ringing etc.). 4) Workload Manager: this gets input from various sensors inside the car and also obtains the current state of the driver (driver gaze, eye movements, braking or accelerating speed etc.). Then it calculates the driver’s cognitive workload. 5) Distributive User Interfaces: as some portion of the risk of accidents depends on surrounding vehicles, these interfaces would securely distribute information about other drivers' characteristics (accident history, number of tickets etc.). This way the driver would be alerted if there is any “risky driver” driving nearby.

CarCOACH [7] was a research project done at MIT in collaboration with Chrysler. The vehicle used in this research was equipped with various sensors, and monitors. The purpose of the research was to provide context-sensitive assistance to neophyte drivers in driving school; it gathers data from various sensors from the car and based on the current driver's workload situations; the system generates either auditory or tactile feedback. The feedback is positive when the driver performs correct action in a situation (such as turn on signal before turning) and negative feedback if the driver performs an incorrect action. CarCOACH's software architecture consisted of a mediator and a scheduler. The mediator queries data from software agents that obtain data from sensors. Before generating feedback to driver, the mediator consults the scheduler, which ensures that the driver is not presented a feedback when overloaded. The CarCOACH was evaluated in an experiment with 18 participants and three types of feedback conditions (no feedback, continuous feedback, and scheduled feedback). The data analysis revealed that positive feedback improved a driver’s performance and decreased his frustration with continuous and scheduled feedback. Negative feedback decreased a driver’s performance and increased his frustration.

John D. Lee et al. [8], in their research on Collision Warning Design to reduce driver distraction, discusses two warning systems. 1) Graded warning system: in this the warning generated is directly proportional to the severity of condition. 2) Single stage warning system: in this the warning is generated when severity of condition surpasses certain value. They conducted two experiments (40 participants, within subject experimental design). Experiment 1 was conducted to measure driver response to these warning systems and alert modality (auditory or haptic). Experiment 2 was conducted to measure the acceptance level of these warning systems by the driver. After conducting statistical tests (ANOVA) on data from experiments, it was found that drivers stopped early in case of graded warning systems, meaning greater safety when compared to single stage warning. Also, graded haptic warning was more gracefully accepted by drivers than auditory warnings.

Yasuo et al. [9] developed a personal driver assistance system. This driving system fits into individual driving behavior. The system is based on a Bayesian network for detecting unusual behavior of the driver. The system is based on the assumption that to measure unusual behavior of driver, normal driving behavior should be measured first. The system was tested in an experiment in which 4 test vehicles equipped with microphone, sensors, and CCD cameras etc. were driven by 67 participants. Recorded data from these devices was stored in an Oracle database. The system measured the driver behavior "stop at intersection". This could be broken down into series of small steps like release the accelerator pedal, put left foot on the brake pedal, turn on the blinker, onset of braking, and stopping. A Bayesian network was constructed from these small steps. Detection of unusual behavior was measured in terms of time to cross the line. This unusual behavior could then be detected and reported in the form of a warning to the driver.

Ksenia Kozak et al. [10] evaluated Lane Departure Warnings (LDWs) for drowsy drivers. Four Human Machine Interfaces (HMIs) for LDW are as follows: Steering Wheel Torque, Rumble Strip Sound, Steering Wheel Vibration, and Heads-up Display. The aim of the experiment was to calculate driver reaction time and acceptance level to these HMI's. The experiment was conducted in Ford's Virtual Track. Twenty-three sleep deprived drivers participated in the experiment. Their drowsiness was computed using a physiological measure of eye closure (PERCLOS).during the entire period of the experiment. Two types of lane departures were applied: driver initiated and using yaw technology, which did not give any perceptible motion cue to the driver. Two performance measures were selected for evaluation of LDW: 1) reaction time of the driver to LDW, computed from steering wheel movement, and 2) lane deviation, defined as the distance between the outer edge of front tire and the outer edge of lane marking. An ANOVA test revealed that the reaction time of the driver was lower with LDW using steering wheel vibration, in conjunction with steering wheel torque, compared to other HMI's. The rumble strip sound HMI was more acceptable to drivers than other HMI's.

John D. Lee et al. [11] conducted an experiment to determine how real-time feedback about off-road glances affects driver distraction. The feedback was either given on dashboard (vehicle centered) or through an in-vehicle information system (IVIS). Feedback provided on IVIS is useful to minimize the distraction caused by looking away from the road, but feedback given on dashboard could help avoid overloading of multiple feedbacks. The hypothesis was that feedback provided on IVIS would be more efficient than dashboard feedback. The experiment was conducted in a driving simulator using a Mercury Sable vehicle cab. Sixteen young and 13 middle aged drivers drove a simulated vehicle having to follow a front vehicle which was braking periodically. The secondary task was to search for a given text on the IVIS display while driving. Statistical analyses on the data showed that irrespective of age and the location where feedback was displayed, drivers paid less attention to IVIS after receiving feedback about distraction.

Cristy Ho et al. [12] at the University of Oxford studied the effectiveness of unimodal auditory, unimodal vibrotactile, and combined audiotactile warnings given to car drivers prior to front vehicle crashes. Braking response time in each of these types of warnings was compared. The study was done in a simulator located at the Transport Research Laboratory, UK. The hypothesis of the experiment was that the audiotactile signal would improve driver performance compared to the other two types of warnings. The driving task was to follow the front vehicle, which stopped at some points in the simulation, while the radio was playing in the background. ANOVA tests were performed on the data obtained from the simulation. It was observed that the driver had the least response time for braking with audiotactile warnings.

John Keller [13] proposed a technique to compute the workload of a simulated human driver. The technique is based on Multiple Resource Theory [18]. The resource usages of the driver are described by four channels: Visual, Auditory, Cognitive, and Psychomotor (VACP). According to this theory, any action performed by a driver consumes some resources from these channels. For example, entering an address into a navigation system requires visual channel usage to look at the screen, cognitive channel usage to recall the address, psychomotor channel usage to type the address, etc. Keller also suggested VCAP resource usage on a scale of 0-7 for several common actions. When the cumulative workload of any channel at particular time reaches above 7, that channel is considered to be overloaded. Although drivers tend to make cell phone calls while driving, it may cause channel overloads and therefore slow or limit other actions a driver can perform. In the CCDM, we have used Keller's guidelines to determine channel usage of a variety of driving and telematics actions; but we translated the 0-7 scale to a 0-100% scale.

Chip Wood and Joshua Hurwitz at Motorola [14] performed an experiment to determine the effect of suspending cell phone calls on driver performance. A simulator was used for this purpose. Twenty participants were involved in this experiment. Their primary task was to follow a front vehicle that would decelerate and accelerate randomly. Their secondary task was to answer a hands free cell phone call; the conversation over phone was either heated, neutral, or none. The collected data (forward velocity, accelerator pressure, brake pressure, and headway distance) were statistically analyzed to compare heated vs. neutral conversation. This revealed that deceleration and braking performance improved when heated conversation was delayed until the car following event was over. Also, after the experiment, a majority of drivers reported that they preferred to delay the cell phone call while heated conversation was taking place. This motivated us to test two IDA delay modes – static delay and dynamic delay – using simulation experiments.

Andrew Liu and D. Salvucci 2001 [15] discussed a Markov Dynamic Model (MDM) that can be used to predict human driver behavior. Driver’s current actions are used to predict his next course of actions. The MDM can be used in real time to predict the driver's immediate next actions. The MDM was tested in an experiment with 8 participants in a simulated car. The model predicted actions with a recognition accuracy of 95%.

Dario D. Salvucci [16] explored cognitive modeling tools for predicting driver distraction using simulation, three of which are related to the CCDM. 1) ACT-R+ Driver Model: this model is based on the cognitive modeling system ACT-R. Salvucci [16] tested cell phone dialing with hand and cell phone dialing with voice using this model. This work is an example of driving model based simulation experiments that we describe in this thesis. 2) ACT-Simple + Driver Model: this model contains predefined operators like "press-key", "look-at" etc. that get converted into ACT-R production rules automatically. Similarly, the CCDM contains predefined actions such as pressing a key or the brake pedal implemented as C++ classes. Salvucci used this model to compare cell phone dialing distraction for young and aged drivers [16]. 3) CogTool + ACT-Simple + Driver Model: in this model, the user can create task scenarios using HTML interfaces; the CCDM also provides a graphical user interface for the user to specify driving scenarios.

Yuji Takada and Osamu Shimoyama [17] at Nissan Motors conducted an experiment to evaluate the effectiveness of Adaptive Cruise Controller (ACC) and Collision Warning System (CWS) driver assistance systems. For this experiment they used a driving simulator, selected 6 males of mean age 39, and had them follow a leading vehicle that either accelerated or decelerated, while performing the secondary task of adding two numbers. They found that ACC produced less mental workload in drivers compared to CWS.

A study of an intelligent driver assistance system that utilizes force feedback (FF) on the accelerator to alert the driver was done by Winter et al. [19] in 2008. Two different types of force feedbacks, 1D and 2D, were compared in a driving simulator with 22 human subjects. The 1D FF system relies on relative distance and speed of the front vehicle to calculate how much acceleration is needed; the 2D FF system calculates weighted averages of Time Headway (THW) and Time To Collision (TTC) of nearby front vehicles within the sensor range of the simulated vehicle. Their hypothesis was that the 2D FF system would be more reliable and more adaptive to the driver's behavior. It was found that the 2D FF system was more preferred over the 1D FF system by a majority of participants.

In this chapter we discussed (1) literature on a variety of approaches to intelligent driver assistance and providing the driver with alerts, feedback or warning, (2) results of evaluating these approaches with actual human drivers executing driving scenarios in driving simulators in the presence of other distracting secondary tasks, (3) probabilistic models to predict driver behaviors, and (4) the use of cognitive models to simulate the effects of secondary tasks while driving. Our research incorporates similar ideas of intelligent driver assistance such as delaying, preventing or suspending secondary tasks - telematics events that could prove to be distracting - and also employs the methodology of simulation-based comparison experiments as others have done. However, unlike the prior work discussed in this chapter, our research proposes four different intelligent assistance techniques and compares the effects of using or not using these techniques using a series of simulation based experiments. Thus, the research reported in this thesis further advances the methodology of simulation based experimentation to evaluate intelligent driver assistance.

Chapter 3

Cognitive-Computational Driving Model Simulator

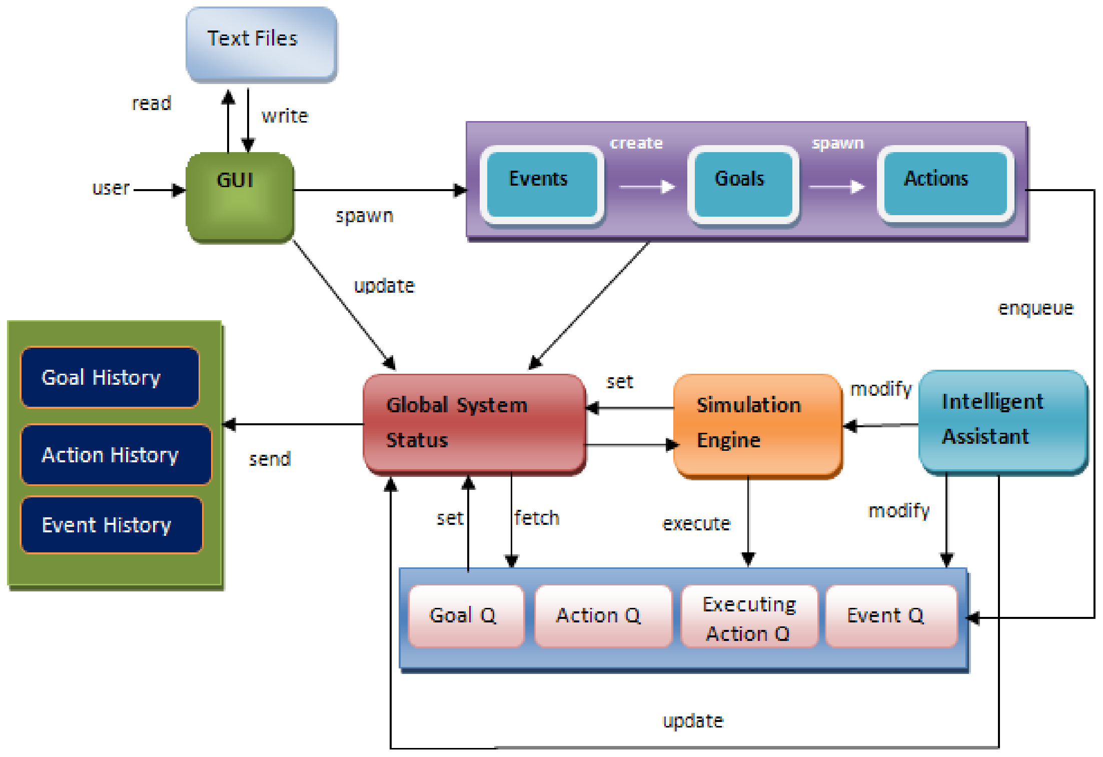


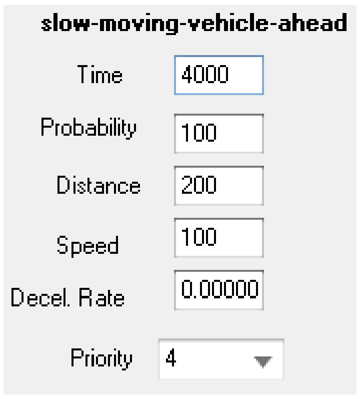
Figure 3.1Driver Modeling Simulation Architecture

Figure 3.1 shows the architecture of the CCDM simulation. At the start of the simulation, the user sees an interface that allows him to schedule a driving scenario of events with various parameters (e.g., a pedestrian is crossing 20 meters ahead). Global System Status (GSS) keeps track of all simulation variables and parameters. The software models a simulated driver in a simulated vehicle, with vehicles possibly in front or to the left or right, driving on a single or multi-lane road, ramp or highway. The events scheduled by the user subsequently spawn driver goals when they are executed (e.g., a cell phone ring event will spawn the driver goal of answering the phone), and update the values of various Global System Status variables. The goals are pushed into a goal queue; when goals start executing, they spawn driver actions. Actions are pushed into an action queue, and in each cycle of the simulation all actions that can be executed by the driver subject to resource constraints on seven channels (cognitive, visual, auditory, speech, and motor: left and right hand and foot) are moved to the executing actions queue.

Within each simulation cycle, the simulation engine executes all driver goals in the goal queue and all actions in the executing actions queue as well as updates the position, speed and other parameters of the simulated and other vehicles on the roadway before advancing the simulation clock. Thus, the system simulates concurrent driver goals and simultaneous driver actions inside a sequential loop. The Intelligent Assistant continuously monitors activities of the driver and external traffic conditions in order to intervene in various ways as described later in this thesis.

Events

In the current version of CCDM, there are 24 different events that can be simulated. A typical event on the graphical user interface looks like Figure 3.2. Name of the above event is "Event slow moving vehicle ahead." The textboxes are parameters of the event. Time specifies the simulation clock value at which the event will occur in the simulation. Probability is an integer between 0 and 100 (probability of occurrence of an event).


slow-moving-vehicle-ahead

Time	4000
Probability	100
Distance	200
Speed	100
Decel. Rate	0.00000
Priority	4

Figure 3.2 Event on the Interface

Distance specifies how far ahead of the vehicle the event will occur. Speed is the speed of the front vehicle. Deceleration rate is rate at which the front vehicle is reducing its speed. Priority is an integer between 1 and 4, with 4 being an emergency event. As this example shows, each of the 24 events can be specified using a set of parameters specific to that event.

Each event is an instance of the event class, which has the following parts:

pre-conditions: Global System Status variable values required for this event to actually occur; if these values are not present in the Global System Status, this event will not be simulated; instead it will be moved to the Event History.

pre-effects: Global System Status variable values adjusted when the event starts executing, to indicate that this event is being executed.

body: Specification of how the event should be executed including the driver goal(s) triggered by the event.

post-effects: GSS variable value changes required to indicate the effects of this event finishing. After the post-effects are processed, the event is moved from the event queue to event history.

Goals

When an event is executed, it spawns driver goal(s) that represent intention in the driver's mind to perform some actions in response to an event. Each goal is an instance of the goal class, which has the following parts:

parent-goal: Inserted by the goal that creates this goal instance.

scheduled-starting-time: Inserted when this goal instance is created.

actual-starting-time: Inserted when this goal instance becomes active.

body: Script of the goal in terms of sub-goals, actions and logic.

status: Inserted by simulation engine into instances of this goal. Status can be scheduled, executing, success, failure or interrupted.

post-effects: GSS variable value changes required to indicate the effects of completing this goal with success or failure; Post-effects also specify the interruption and removal of any sub goals and actions that this goal created from the goal queue, action queue and current actions (this is the queue of currently executing actions) to the respective histories. The last step of post-effects is for the goal to move itself from goal queue to goal history.

interrupt: This field specifies that when this goal is interrupted, all sub goals and actions that this goal created should also be interrupted and removed from the goal queue, action queue and current actions to the respective histories. Then this goal moves itself from goal queue to goal history.

Actions

An Action executes at the lowest level in the hierarchy of the CCDM. It inherits certain parameters from its parent Basic Action class. Following are the parts of an action:

parent goal: Inserted by the goal that creates this action object.

priority: Inherited from and inserted by the parent goal.

previous-action: This encodes dependencies such as this action can be done only after another action is completed.

next-action: This encodes dependencies such as another action has to be done immediately after this action.

parallel-action: This encodes dependencies such as this action has to be done in parallel with another action, so either both or neither will be scheduled for execution by the simulation engine.

status: Inserted by simulation engine as it schedules actions. Status can be scheduled, executing, success, failure or interrupted.

post-effects: GSS variable value changes required to indicate the effects of completing this action with success or failure.

interrupt: GSS variable value changes required to indicate the effects of this action being interrupted during execution.

resource-requirements

vision: Extent of vision channel usage of this action as a percentage.

hands: If a percentage is specified here but left and right hand percentages are empty, this indicates hand use requirement with either hand; when this action is scheduled, the simulation engine will choose whichever hand is available.

left-hand: Extent of the specific hand usage required for this action as a percentage.

right-hand: Extent of the specific hand usage required for this action as a percentage.

left-foot: Extent of the specific foot usage required for this action as a percentage.

right-foot: Extent of the specific foot usage required for this action as a percentage.

auditory: Extent of the use of auditory resources required for this action as a percentage.

speech: Extent of the use of speech production resources required for this action as a percentage.

cognition: Extent of mental processing required for this action as a percentage.

Simulation Engine

A discrete event simulation engine is the kernel of the CCDM. It performs various tasks including the following: 1) instantiates user input forms, 2) interfaces with Microsoft Excel and writes simulation data into Excel after each simulation cycle, 3) executes goals, events and actions, 4) updates simulated vehicle and other vehicle locations, speeds etc., 5) detects accidents, and 6) executes Intelligent Assistance. These tasks are done in every iteration of a simulation loop, after which the simulation clock is incremented by a user-specified interval that defaults to 5 milliseconds.

Global System Status

This serves as the globally accessible storage of all information for CCDM. With each simulation cycle, either information from Global System Status is read or it is updated by goals, actions, events, or the simulation engine.

Intelligent Driver Assistance

Intelligent Driver Assistance (IDA) consists of four modes of operation, namely: 1) safe mode, 2) dynamic delay, 3) static delay and 4) cognitive resource limits. The description of each mode can be found in the chapters of this thesis that discuss experiments with each mode.

Key Features of the CCDM

1) **Safe distance:** The simulation uses a "two-second rule" to maintain a safe distance between the simulated vehicle and the vehicle in front. If the vehicle is less than two seconds away at its

current speed from the vehicle or an obstacle in front, a braking action is initiated or re-initiated and the deceleration rate of the simulated vehicle is re-calculated (thus simulating harder braking) up to the maximum deceleration rate of the vehicle in order to slow down (if what is in front is a vehicle that is moving) or to stop (if what is in front is a stationary obstacle). This "two-second distance" is calculated in meters as "(current speed of the vehicle in KMH) * 0.55555" for a stationary obstacle in front or as "(speed difference between simulated vehicle and the moving obstacle in front in KMH) * 0.55555" for a moving obstacle in front.

2) **Stochastic modeling of individual variability:** There is quite a bit of individual variability in driving. Different drivers of the same age may behave differently in the same driving situation. The same driver may behave differently in different driving situations. The same driver may also behave differently in the same driving situation at different times. To model these variations, the simulation randomizes a variety of parameters in order to simulate the individual variability of drivers over time, and to account for driver age, time of day and weather conditions. The following perceptual and motor parameters are adjusted by a random percentage drawn from the range [-5%, +5%] using a uniform distribution:

Braking behavior: deceleration rate of the vehicle.

Accelerating behavior: acceleration rate of the vehicle.

Visual perception delay of the driver.

Auditory perception delay of the driver.

Decision making delay of the driver.

Time to release a pedal by the driver.

Time to operate a lever by the driver.

Time to press a switch/button/key by the driver.

Time to turn the wheel by one degree by the driver.

3) **Stochastic modeling of driver age:** The literature indicates that perception delays and reaction times increase with age. The simulation uses plausible values that we determined or mean values collected from the literature as default values of the following parameters for a young driver. These values are then increased by a random percentage drawn from the range 0-5% using a uniform distribution for an adult driver or by a random percentage drawn from the range 0-10% using a uniform distribution for an old driver.

Visual perception delay of the driver.

Auditory perception delay of the driver.

Decision making delay of the driver.

Time to release a pedal by the driver.

Time to operate a lever by the driver.

Time to press a switch/button/key by the driver.

Time to turn the wheel by one degree by the driver.

Modeling the influence of weather and time of day.

4) **Stochastic modeling of the influence of weather and time of day:** The simulation models two times (day/night) and four kinds of weather (clear, rainy, foggy, or snowy). Visibility is less at night, so night time driving will have the effect of decreasing visibility, which is modeled in the simulation by increasing the visual perception delay by 0-5%. Rainy/snowy/foggy weather increases visual perception delay by 0-10%.

Rainy weather also decreases braking efficiency. This is modeled in the simulation by reducing the deceleration produced by braking by 0-5%. Snowy weather further decreases braking efficiency. This is modeled in the simulation by reducing the deceleration produced by braking by 0-10%. In all cases, the actual increase or decrease of a parameter is computed as a percentage drawn from the given range using a uniform distribution.

5) **Road characteristics and traffic conditions:** The simulation models three types of roads: regular, highway and ramp. The system interface allows the user to specify the number of lanes and speed limit of the type of road the simulated vehicle will be traveling on at the start of a simulation. Events allow the specification of lanes and speed limits for different types of roads that the vehicle may enter and travel on subsequently.

The traffic density parameter is specified as a %, which is used as a probability by the simulation to place a vehicle to the right, left or front of the simulated vehicle. This probability is recomputed each minute of the simulation. However, if the simulated scenario includes the explicit event placing or removing vehicles to the left, right or front of the simulated vehicle, once the first such event is executed, the traffic density parameter is no longer used by the simulation to decide whether there should be a vehicle to the left, right or front of the simulated vehicle. That is, the event "other vehicles" supersedes the traffic density parameter specified at the start of a simulation.

Baseline of Resource Consumption for Normal Driving

Based on the research of John Keller [13], who specifies a 0-7 workload scale for Visual, Auditory, Cognitive, and Psychomotor (VACP) resource consumption for use in simulation modeling, we have assigned VACP resource needs to various actions performed by the driver while driving.

These resource requirements are incorporated in the corresponding action class definitions. In CCDM, VACP resource consumption by actions is specified on a percentage scale of 0 to 100 and their consumption by actions is determined based on John Keller’s recommendations.

Experiments described later in this thesis use certain baseline values of resource consumption to determine whether the simulated driver is overloaded in any of the resource channels: vision, hearing, speech, cognition and motor.

These baselines values reflect the driver's resource usage when normal driving is simulated. The following table illustrates this resource usage when a simple driving task is simulated for 10000 milliseconds.

Time	Auditory	Cognition	Left foot	Left hand	Right foot	Right hand	Speech	Vision
2	15	128	0	0	100	100	0	100
4	15	128	0	0	100	100	0	100
6	15	128	0	0	100	100	0	100
8	15	128	0	100	100	0	0	100
10	15	128	0	100	100	0	0	100

Table 3.1 Workload Output for Normal Driving

This pattern stays constant for regular driving even for longer simulations. From the above table we can deduce that for normal driving task, the baseline resource usage by the simulated driver is as follows:

Auditory	15
Vision	100
Cognition	128
Either hand	100
Right foot	100
Left foot	0
Speech	0

Table 3.2 Baseline Resource Usage for All Channels

If the driver performs any additional task like making a cell phone call while driving, then the resource usage may go beyond these baseline values; but, this does not necessarily mean that the driver is driving at high risk, rather only that his resource channels are overloaded or that his workload is above the baseline for routine driving.

Chapter 4

Experiment: Limited Cognitive Resources vs. Unlimited Cognitive Resources

In this experiment, we compared the effectiveness of limiting the driver's use of visual, auditory, speech and cognitive resources through the Intelligent Driver Assistance subsystem. Although current technology in cars is not capable of measuring a driver's cognitive workload while driving, such technology may become available in future; we simulated such a futuristic IDA capability [6] in which the system would be able to track the usage of the driver's visual, auditory, speech and cognitive channels and automatically delay driver actions that would cause these channel utilizations to go beyond prescribed limits.

For the simulation experiment, we used a scenario with event "stop at a signal", event "enter a destination address into a navigation device", and event "answer a ringing cell phone". The idea behind this experiment was to find out if an IDA system that is able to prevent the driver from doing actions that will overload his visual, auditory or cognitive channel is used, what effect will such a system have on driving safety when the driver has to deal with an emergency while operating a telematics device (or starts to operate a device while dealing with an emergency).

4.1 Operation of the Cognitive Resource Limit IDA Mode

Cognitive Resource Limit applies to the following four resource channels:

1) Vision

2) Auditory

3) Speech

4) Cognition

When we set these channels' limits to x% in a simulation run, this means that the IDA would not allow any actions to consume these channel resources beyond x%. Any action that causes a channel usage to exceed x% will not be executed unless and until that channel usage decreases sufficiently below x% to accommodate the action's need of that channel resource. This ensures that no resource channel of the driver is overloaded when any emergency driving goal is executing.

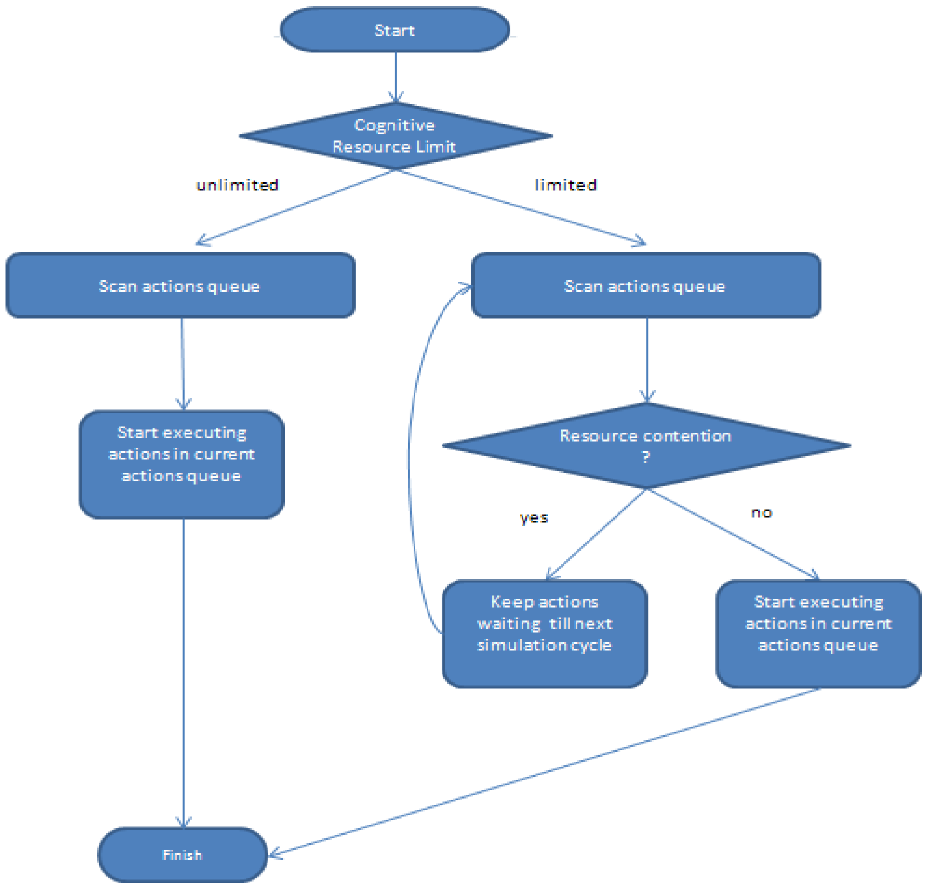

Start
Cognitive Resource Limit
unlimited
limited
Scan actions queue
Scan actions queue
Start executing actions in current actions queue
Resource contention ?
yes
no
Keep actions waiting till next simulation cycle
Start executing actions in current actions queue
Finish

Figure 4.1.1 Flowchart of Limited Cognitive Resources Mode

The net result is that the intelligent assistant will prevent any other driver goal or action from executing in parallel with the actions of the emergency driving goal when any of the four resource channels of the driver are already overloaded. When the resource channel limits are set to unlimited or a very large percentage, the intelligent assistant will not prevent any other goals or actions from executing at the same time as emergency goals. This allows us to analyze how the driver executes various actions, and the driver's workload, when there are no resource constraints and compare that with a simulation with set resource constraints.

4.2 Design of the Simulation Experiment

Simulation		**Unit**
Simulation Length:	25000	ms
Driver Age:	Old	
Time of Day:	Night	
Weather:	Foggy	

Table 4.2.1 Simulation Parameters

As shown in table 4.2.1, we selected simulation length as 25000 milliseconds, an old driver and foggy weather for this experiment. When simulation clock reached 2 seconds, the event "stop at signal" was fired with parameters mentioned in table 4.2.2; this subsequently created a driver goal to stop. When simulation clock reached 3 seconds, events shown in table 4.2.3, and 4.2.4 were simulated.

As shown in figure 4.2.1, we selected resource limit values to be baseline values of the four channels – the usage levels of these channels when the simulated driver is driving normally and doing nothing else. Note that the baseline value for cognition is already at 128% - this is not an anomaly, and matches the Visual, Auditory, Cognitive and Psychomotor theory of workload

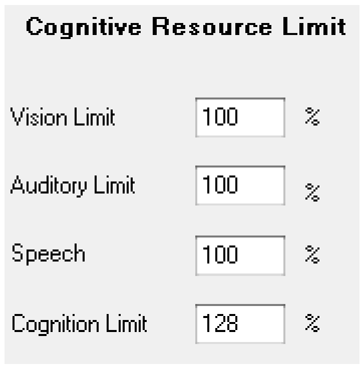
[13]. Double the baseline values are what we consider to be a significant overload with the potential to make driving unsafe. Simulation runs with channel limits set to baseline values were then compared against runs of the same simulated scenario but with no limits on channels.

Event stop at signal		Unit
Time:	2000	ms
Probability:	100	
Turn direction	Right	
Distance	130	m
Priority:	4	

Table 4.2.2 Event 1 Parameters

Event enter GPS address		Unit
Time :	3000	ms
Address length:	7	
Probability:	100	
Priority:	3	

Table 4.2.3 Event 2 Parameters


Cognitive Resource Limit

Vision Limit	100	%
Auditory Limit	100	%
Speech	100	%
Cognition Limit	128	%

Figure 4.2.1 Baseline Driving Resource Limits

Event cell phone ring		Unit
Time :	3000	ms
Duration:	10000	ms
Probability:	100	
Priority:	3	

Table 4.2.4 Event 3 Parameters

4.3 Output Data

Each time the simulation is run, two outputs are generated as excel files. The first output file describes workload output, event history, and goal history. Workload output illustrates how much usage of the auditory channel, the cognition channel, left and right foot, left and right hand, the speech channel, and the vision channel of the simulated driver were required during the simulation run. The event history shows which event occurred at what time. The goal history shows information such as which driver actions or goals were created during the simulation, how long they lasted, and whether these were successful or not. The second output file describes raw data from the entire simulation. It shows the status of the simulated vehicle including speed, location, deceleration rate, acceleration rate, current lane etc. for every five milliseconds of the simulation clock. The data from these two files were analyzed using 10 metrics for measuring the effects of limiting cognitive resources.

4.4 Results

We analyzed the collected data in terms of the following 10 metrics to compare the effects of limiting or not limiting visual, auditory, speech and cognitive resource usage. Table 4.4.1 provides mean values and standard deviations of these metrics, and p-values resulting from a two-tailed unequal variance t-test comparison of the means for each metric.

	Metric 1 (ms)	Metric 2 (ms)	Metric 3 (ms)	Metric 4 (ms)	Metric 5 (ratio)	Metric 6 (ratio)	Metric 7 (%)	Metric 8 (sec)	Metric 9 (%)	Metric 10 (sec)
Unlimited Cognitive resources mean (SD)	13107 (478.9352)	12538 (173.880)	14538 (173.88)	11727 (2035.193)	0.870504 (0.299475)	0.887998713 (0.061250)	259.5 (41.0589)	13.4 (0.96609)	280 (32.8937)	11.4 (0.966092)
Limited cognitive resources mean (SD)	11415 (2002.75)	NA (NA)	10245 (0)	1984.5 (14.42413)	0.102229 (0.039993)	2.15 (0.0853610)	100 (0)	6.4 (1.577621)	128 (0)	2 (0)
t-test p-value	0.053754	NA	NA	0.0000001	0.000021	NA	0.0000006	0.0000000047	0.000000141	0.0000000001

Table 4.4.1 Experimental Data

Metric 1: Safety: Attention

This is the total time of driver gaze that was related to driving, i.e., on the windshield or one of the three mirrors or out of left/right window. More time on driving gaze suggests better attention to driving related tasks. Limiting cognitive resources seems to decrease the time driver's gaze was related to driving, but this difference did not reach statistical significance ($p>0.05$). This suggests that resource usage did not significantly affect (decrease or increase) driving related gaze time; so no conclusion can be reached regarding safe driving.

Metric 2: Safety: Time on Safety-Critical Goal

This is the total time needed for safety-critical goal accomplishment; lesser the time required, more efficient and safer the driving. As event "stop at signal" failed to finish within the length of the simulation in the presence of limited cognitive resources, we were unable to evaluate Metric 2.

Metric 3: Safety: Time on Driving Actions

This is the total time needed for the accomplishment of all (not just safety-critical) driving related actions. Ideally this should be the same in both modes, limited and unlimited cognitive resources, because an emergency driving goal always has priority over non-emergency goal and is never terminated by any other goal. So in this experiment, the total time on driving actions would have been the total time on actions related to the goal of stopping at a signal, which should be the same in both conditions. However, the emergency goal did not complete its execution within the predetermined simulation time because of limited cognitive resources. Hence, we cannot compare the mean values of Metric 3 between the two conditions.

Metric 4: Safety: Reaction Time for Safety-critical Actions

This is the delay between when any emergency driving-related action is created and when it actually starts executing. This metric captures the response delay or reaction time of the simulated driver. We can see from the mean values for Metric 4 that the reaction time with limited cognitive resources is less compared to unlimited cognitive resources. Also, the p-value is less than 0.05. Hence, limited cognitive resources mode is better in terms of safety according to this metric.

Metric 5: Safety & Efficiency: Ratio of Safety-Critical Reaction Time and Telematics Delay for Actions

This is the ratio of the delay between when any safety-critical action is created and when the driver starts executing it and the delay between when any telematics device related action is created and when the driver starts executing it. Ideally this should be less than 1 and small because we want the driver to be more responsive to safety-critical driving actions than actions on a telematics device such as a navigation device. In case of limited cognitive resources, this ratio is smaller ($p < 0.05$) compared to unlimited cognitive resources. Hence, limited cognitive resources mode is better.

Metric 6: Safety: Interface Actions to Driving Actions Ratio

This is the ratio of total number of telematics-device related actions to total number of driving related actions. Ideally, this should be less than 1 and small. But because the emergency driving goal did not finish within the predetermined simulation time when resources were constrained, driving actions were not completed. Therefore, the total number of driving actions is less than interface actions, and this ratio is greater than 1 in case of limited cognitive resources. However, we cannot compare the mean values of Metric 6 between the two conditions.

Metric 7: Safety: Visual Perception Overload Extent

This is the maximum load of the visual channel above the baseline value of 100%. With unlimited cognitive resources, visual perception overload extent is very high ($p<0.05$) compared to the limited resources mode. Hence limited cognitive resources mode is better according to this metric.

Metric 8: Safety: Visual Perception Overload Time

This is the total time for which the driver's visual channel load is greater than or equal to 100%. Clearly, with limited cognitive resources, mean time for which the visual channel load is greater than 100% as well as the overload extent are significantly less compared to unlimited cognitive resources. Hence, with respect to this metric, the limited cognitive resource mode is better.

Metric 9: Safety: Cognitive Processing Overload Extent

This is the maximum load of the cognition channel above the baseline value of 128%. The extent of cognitive overload is higher ($p<0.05$) with unlimited cognitive resources mode than with its counterpart mode. Hence, limited cognitive resources mode improves driving safety by keeping cognition load at its baseline level.

Metric 10: Safety: Cognitive Processing Overload Time

This is the total time for which the cognitive channel load is greater than or equal to the baseline value of 128%. Clearly, with limited cognitive resources, mean time for which the visual channel load is greater than 100% as well as the overload extent of cognition are significantly less compared to unlimited cognitive resources. Hence, with respect to this metric, the limited cognitive resource mode is better.

4.5 Conclusion

From the results of the above metrics we can conclude that the limited cognitive resources mode is better than the unlimited cognitive resources mode with respect to Metric 4, Metric 5, Metric 7, Metric 8, Metric 9, and Metric 10. But these results may be skewed by the fact that the safety-critical driving goal of stopping at a signal was not completed by the time the simulation ended. The true effect of this mode could be observed if we could simulate a scenario of sufficient length in which actions of goal stop at signal wait until resources become available, execute, and thus complete the goal. This requires additional simulation experiments in future.

Chapter 5

Experiment: Dynamic Delay on vs. Dynamic Delay off

In this experiment, we compared the effectiveness of the dynamic delay capability of the Intelligent Driver Assistance subsystem. We used the event "obstacle in lane ahead" and event "enter GPS destination by voice" for this purpose. The idea behind this experiment was to find out if an IDA system is able to temporarily prevent the driver from operating a telematics device while engaged in an emergency maneuver until that maneuver is completed, what effect such a system will have on driving safety. One way to implement this capability is to install a variety of sensors inside the car and collect data from them; once the data is collected, we can supply it to a predictive Bayesian network [7] to determine if the driver is performing an emergency maneuver or not, and if so, employ an IDA to delay all non-emergency tasks involving telematics devices by modifying their operation until the emergency finishes.

5.1 Operation of Dynamic Delay

Dynamic delay is currently implemented for the following telematics devices/events:

1) Cell phone rings.
2) Driver enters destination address on a navigation device using touch.
3) Driver enters destination address on a navigation device using voice.
4) Navigation device issues voice instructions to the driver.
5) Driver makes a cell phone call.

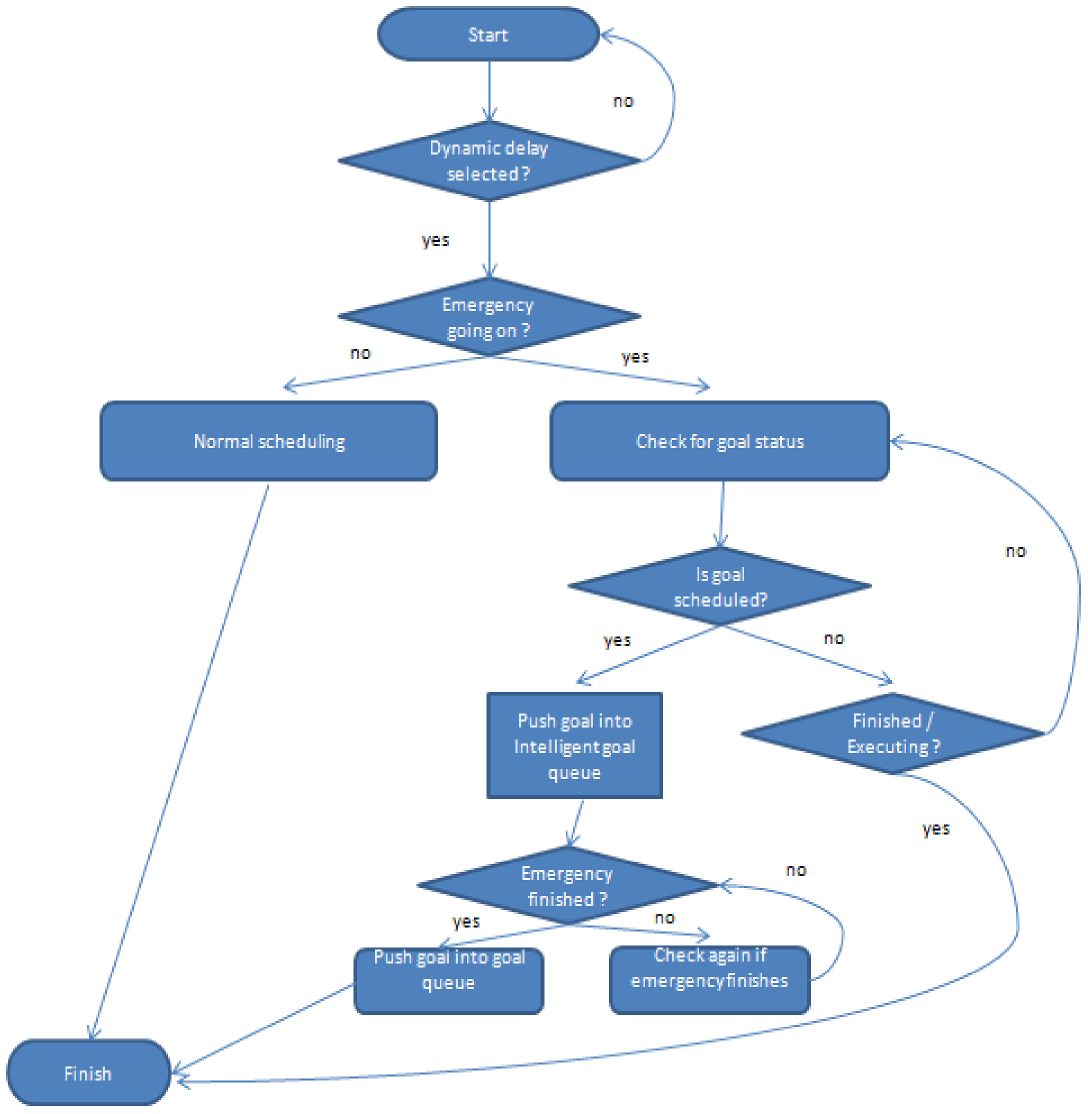


Figure 5.1.1 Flowchart of Dynamic Delay Mode

Figure 5.1.1 shows a flowchart of the operation of dynamic delay mode. When an emergency goal is executing, and if the dynamic delay mode is turned on for any of the above events, intelligent assistance starts working. It checks the goal queue to find out if any of the above events has spawned a goal with its status being scheduled. If it finds that the goal spawned by any of the above events is already executing or finished executing, then the dynamic delay has no effect on the simulation run. Otherwise, if the goal status is scheduled, the IDA removes that goal from the goal queue and pushes it into an IDA goal queue. The goal resides in this goal queue until the emergency goal either fails or finishes executing with success. After the emergency maneuvering is over, the IDA pushes the goal back into the goal queue for normal execution. This simulates the driver not being notified of distracting events such as a ringing cell phone or being prevented from operating a device such as a GPS navigation device until any emergency maneuvers are over, and hence driving safety is likely to be enhanced.

5.2 Design of the Simulation Experiment

Simulation		Unit
Simulation Length:	30000	ms
Driver Age:	Old	
Time of Day:	Night	
Weather:	Snowy	

Table 5.2.1 Simulation Parameters

For this experiment, we selected the two events shown in table 5.2.2 and table 5.2.3, and the simulation parameters are shown in table 5.2.1. When the simulation clock reached 4000 milliseconds, an emergency event, "obstacle in lane ahead", was fired with the obstacle located 250 meters ahead of current location of the vehicle in the same lane. When the simulation clock reached 6000 milliseconds, a priority 2 event, "enter GPS destination by voice", was added to the event queue.

Event obstacle in lane ahead		Unit
Time :	4000	ms
Distance:	250	m
Probability:	100	
Priority:	4	

Table 5.2.2 Event 1 Parameters

Event enter GPS destination by voice		Unit
Time:	6000	ms
Duration:	500	ms
Probability:	100	
Recognition rate:	100	
Address length:	9	
Priority:	2	

Table 5.2.3 Event 2 Parameters

Figure 5.2.1 shows the list of available events that can be delayed dynamically in the current version of CCDM. We ran the simulation 10 times with dynamic delay on and 10 times with dynamic delay off with exactly the same parameters.

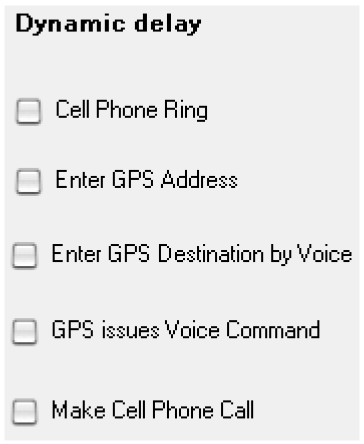


Figure 5.2.1 Possible Events with Delay Functionality

5.3 Output Data

The output files generated by the simulation runs are the same as those described in Chapter 4.

5.4 Results

We analyzed the collected data in terms of the following 10 metrics to compare how the simulated driver dealt with the two events with dynamic delay selected and un-selected. Table 5.4.1 provides mean values and standard deviations of these metrics, and p-values resulting from a two-tailed unequal variance t-test comparison of the means for each metric.

	Metric 1 (ms)	Metric 2 (ms)	Metric 3 (ms)	Metric 4 (ms)	Metric 5 (ratio)	Metric 6 (ratio)	Metric 7 (%)	Metric 8 (sec)	Metric 9 (%)	Metric 10 (sec)
dynamic delay off mean (SD)	27188 (396.2126)	8937.5 (137.34081)	29050 (0)	1500 (42.1637)	0.186936 (0.00963)	0.287086581 (0.0080140)	164.8 (22.76839)	27.6 (0.84327)	637.6 (354.2488)	26 (4.988876)
dynamic delay on mean (SD)	26850 (378.3884)	8985 (158.78007)	29050 (0)	1535 (55.8271)	0.224498 (0.01376)	0.287913167 (0.0086857)	166.7 (35.38690)	27.4 (0.966091)	970.1 (40.0928)	27.4 (0.966091)
t-test p-value	0.06684	0.483672478	NA	0.132342	0.00000255	0.82746	0.888314	0.627944	0.015811	0.404732

Table 5.4.1 Experimental Data

Metric 1: Safety: Attention

This is the total time of driver gaze that was related to driving, i.e., on the windshield or one of the three mirrors or out of the left/right window. More time on driving gaze suggests better attention to driving related tasks. As we can see from the table, there is not a significant difference in the time of driver gaze related to driving between the two conditions ($p>0.05$). Hence, we conclude that the total time of driver gaze is unaffected by dynamic delay.

Metric 2: Safety: Time on Safety-Critical Goal

This is the total time needed for safety-critical goal accomplishment; lesser the time required, more efficient and safer the driving. As the mean time to finish the goal "avoid stationary obstacle" is not significantly different ($p>0.05$) in either of the modes, we conclude that the total time of safety-critical driving goal accomplishment is unaffected by dynamic delay.

Metric 3: Safety: Time on Driving Actions

This is the total time needed for the accomplishment of all (not just safety-critical) driving related actions. We expect that this should be the same in both modes because dynamic delay should only affect emergency maneuvers. Data shows that all driving related actions take exactly the same time to finish their execution whether the dynamic delay mode is on or off.

Metric 4: Safety: Reaction Time for Safety-Critical Actions

This is the delay between when any emergency driving-related action is created and when it actually starts executing. This metric captures the response delay or reaction time of the simulated driver. If dynamic delay is effective in influencing driver response time, we should see that the mean value of this metric is lower when dynamic delay is on. We can see from the t-test that $p>0.05$, so there is no significant difference in reaction times of safety-critical actions between the two modes. Hence, we conclude that driver reaction time is unaffected by dynamic delay.

Metric 5: Safety & Efficiency: Ratio of Safety-Critical Reaction Time and Telematics Delay for Actions

This is the ratio of the delay between when any safety-critical action is created and when the driver starts executing it and the delay between when any telematics device related action is created and when the driver starts executing it. Ideally this should be less than 1 and small because we want the driver to be more responsive to safety-critical driving actions than actions on a telematics device such as a navigation device. We expected this ratio to be smaller when dynamic delay was on. However, the mean ratio is significantly smaller ($p<0.05$) when the dynamic delay is off than when it is on. The reason for this ratio to become large when dynamic delay was on is the following. The safety-critical reaction time, i.e., numerator of the ratio, was almost the same when dynamic delay was on and when it was off. But telematics reaction time, i.e. denominator of the ratio, was decreased by the dynamic delay mode because the telematics goal was waiting in the IDA queue and none of its actions were executing when the safety-critical driving maneuver was going on; when that was finished, the telematics actions could be quickly accomplished because resources were available and therefore the delay for telematics actions decreased. Thus, the dynamic delay mode did not decrease safety-critical driving reaction time, but it improved telematics action delay. But this improvement was negated by the delay in telematics goal accomplishment as the goal waited in the IDA queue while safety-critical driving actions were executing. This leads to the conclusion that this metric is inconclusive with respect to the two conditions.

Metric 6: Safety: Interface Actions to Driving Actions Ratio

This is the ratio of the total number of telematics-device related actions to the total number of driving related actions. Ideally, this should be less than 1 and small. These ratios in both conditions (dynamic delay on and off) are not significantly different ($p>0.05$). Hence, we conclude that the numbers of interface-related and driving-related actions are unaffected by dynamic delay.

Metric 7: Safety: Visual Perception Overload Extent

This is the maximum load of the visual channel above the baseline value of 100%. We can see that there is not much difference with respect to visual perception overload extent ($p>0.05$) in both conditions. Hence, we conclude that the extent to which the driver's visual channel is overloaded is unaffected by dynamic delay.

Metric 8: Safety: Visual Perception Overload Time

This is the total time for which the driver's visual channel load is greater than or equal to 100%. There is no significant difference between the means in the two conditions. Hence, we conclude that the extent to which the driver's visual channel is overloaded is unaffected by dynamic delay.

Metric 9: Safety: Cognitive Processing Overload Extent

This is the maximum load of the cognitive channel above the baseline value of 128%. The extent of cognitive overload is significantly higher ($p<0.05$) when dynamic delay is on Hence, we conclude that the driver's cognitive channel is overloaded by dynamic delay.

Metric 10: Safety: Cognitive Processing Overload Time

This is the total time for which the cognitive channel load is greater than or equal to the baseline value of 128%. Cognitive processing overload time was not significantly different between when dynamic delay was on and when it was off ($p>0.05$). Hence, dynamic delay has no effect on the time for which the driver's cognitive resources are overused.

5.5 Conclusion

From the results of the above comparisons we can conclude that for the scenarios tested, the dynamic delay off mode is better than the dynamic delay on mode with respect to safety and cognitive resource overload. Thus, this particular IDA capability does not seem to assist the driver.

Chapter 6

Experiment: Safe Mode on vs. Safe Mode off

In this experiment, we evaluated the effectiveness of the safe mode of the Intelligent Driver Assistance subsystem. We used a scenario with the events "driver makes a cell phone call" and "vehicle cutting in front" for this purpose. Our aim was to measure whether and how safe mode assists safe driving, either by delaying interruptions by in-vehicle telematics devices or aborting driver interactions with such devices if already going on. The capability this simulation models is that of a futuristic IDA that is capable of detecting, using sensors, whether the driver is engaged in a safety-critical maneuver and if so either disabling any telematics devices the driver is interacting with or delaying any pending interruptions (such as ringing) from such devices until the driving maneuver is completed.

6.1 Implementation of Safe Mode Simulation

Figure 6.1.1 shows operation of safe mode. If safe mode is on, whenever driver actions corresponding to a safety-critical event (such as the front vehicle suddenly stopping) starts executing, the IDA would check to see if any lower priority actions (such as those corresponding to operating a telematics device) are currently executing. If so, these actions are interrupted, effectively simulating the temporary disabling of the device.

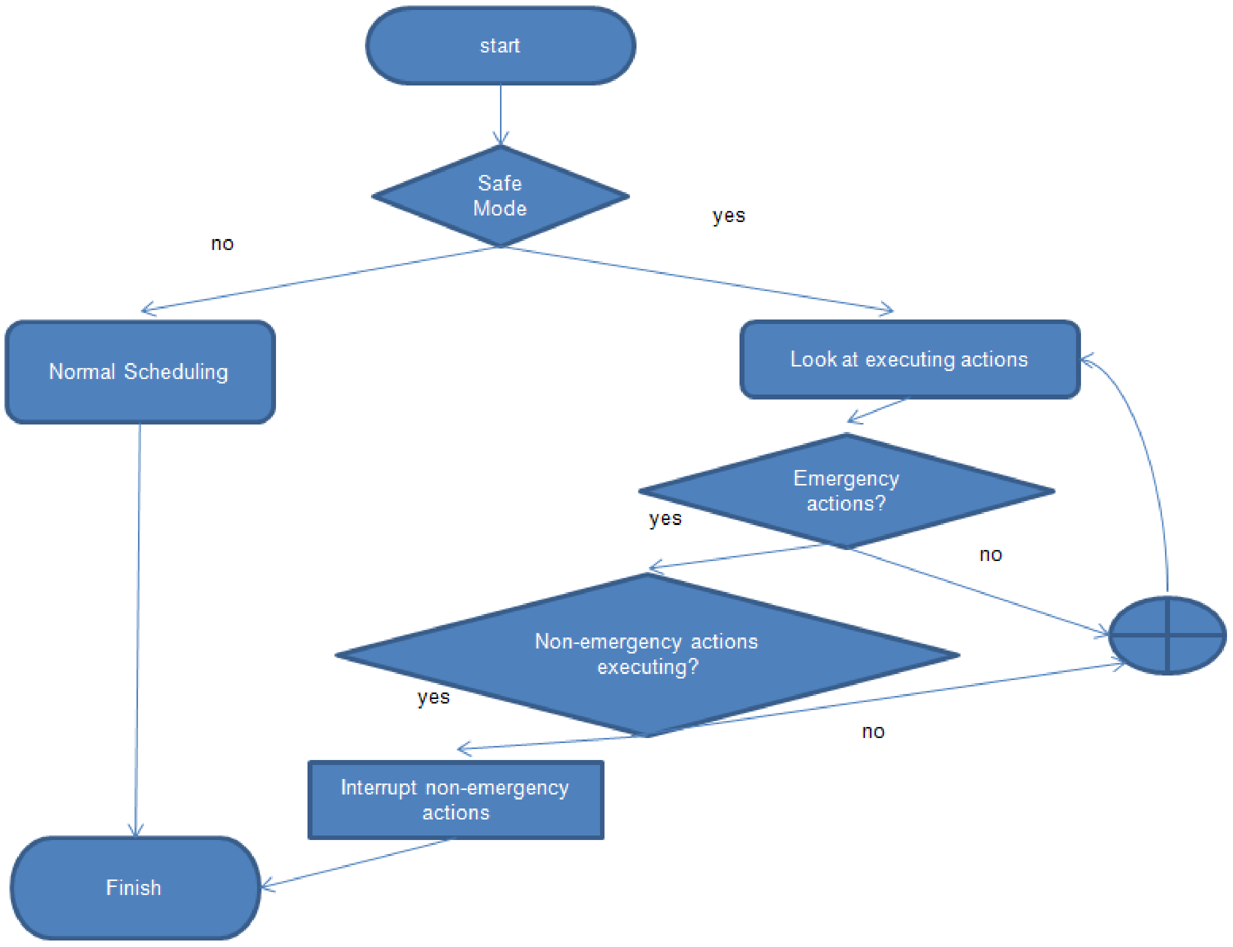

start
Safe Mode
no
yes
Normal Scheduling
Look at executing actions
Emergency actions?
yes
no
Non-emergency actions executing?
yes
no
Interrupt non-emergency actions
Finish

Figure 6.1.1 Flowchart of Safe Mode Operation

If such actions are scheduled, it would delay these actions until all safety-critical driving actions have finished executing (i.e., delaying a device interruption). If safe mode is off, then the simulation would execute all driver actions as long as cognitive resources are available.

6.2 Design of the Simulation Experiment

Table 6.2.1 summarizes the simulation parameters used in this experiment. In order to evaluate the safe mode, it is important to compare a simulation scenario in which the safe mode is on with the same scenario with the safe mode off, with the rest of the simulation parameter values kept constant. In particular, there are three external factors that are modeled in the simulation.

Simulation		Unit
Simulation Length:	25000	ms
Driver Age:	Old	
Time of Day:	Night	
Weather:	Snowy	

Table 6.2.1 Simulation Parameters

The first is the driver's age, which has three options: young, adult, or old. Whenever the age parameter is changed from young to adult or adult to old, there will be a random increase in the range 0-5% in visual perception delay and object operation delay of the driver. Second is the Time of Day: either day or night. There will be another random 0-5% increase of the driver's visual perception delay for night driving. The third is weather conditions, which include sunny, rainy, snowy, and foggy. For weather changes from sunny to rainy, snowy, or foggy, there will be random 0-5% increases in the visual perception delay of the driver.

If the weather is rainy or snowy, the simulation will randomly reduce the vehicle's braking efficiency in the range 0-5%. There is also a [-5%, +5%] randomization in deceleration/acceleration rates as a result of the driver operating the gas/brake pedal, to model human variability. Thus, the simulation behaves in stochastic fashion.

The experiment consisted of simulating a driving scenario with two events:

1) Driver makes a cell phone call.

2) A vehicle suddenly cuts in front.

Event vehicle cutting in front		Unit
Time:	6000	ms
Probability:	100	
Distance:	150	m.
Priority:	4	
Speed:	40	Km/hr.
Deceleration rate:	0.000001	meter/milliseconds^2
Acceleration rate:	0	meter/milliseconds^2

Table 6.2.3 Event 2 Parameters

Event make cell phone call		Unit
Time:	2000	ms
Duration:	12000	ms
Probability:	100	
Priority:	3	

Table 6.2.2 Event 1 Parameters

Tables 6.2.2 and 6.2.3 provide these events' parameters.

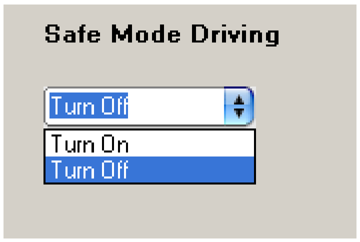


Figure 6.2.1 Safe Mode Driving On/Off Switch

From the above we can see that in the simulated driving scenario, the driver made a cell phone call at 2000 milliseconds into the simulation run and the call lasted for 12000 milliseconds; priority 3 indicates that this was a non-emergency event. When the simulation clock reached 6000 milliseconds, a vehicle cut in front of the simulated vehicle at a distance of 150 meters, decelerating from a speed of 40 km/hour and coming to a stop. Priority 4 indicates that avoiding this vehicle is a safety-critical task; failure to do so would result in an accident. Figure 6.2.1 shows how we can turn on/off safe mode from the CCDM interface.

6.3 Output Data

The output files generated in this experiment are the same as those described in Chapter 4.

6.4 Results

	Metric 1 (ms)	Metric 2 (ms)	Metric 3 (ms)	Metric 4 (ms)	Metric 5 (ratio)	Metric 6 (ratio)	Metric 7 (%)	Metric 8 (sec)	Metric 9 (%)	Metric 10 (sec)
Safe Mode off mean (SD)	22637.5 (1010.234)	10631.5 (2296.688498)	24050 (0)	11099 (4280.483)	1.70709 (0.503574)	0.215703 (0.014294)	175.9 (6.2795965)	20.8 (1.032795)	226 (0)	20.8 (1.032796)
Safe Mode on mean (SD)	23109 (423.8042)	9017.5 (166.8374125)	24050 (0)	1775 (43.20494)	88.75 (2.160247)	0.098993 (0.0371)	172 (0)	20.6 (0.9660917)	204 (0)	20.6 (0.96609)
t-test p-value	0.198378	0.053573668	NA	0.000071558	0	0.000001984	0.081126	0.660079	NA	0.660079

Table 6.4.1 Experimental Data

We analyzed the data in terms of the following 10 metrics to compare how the driver and driving are affected when the driver is dealing with the two events with the safe mode on and with the safe mode off. Table 6.4.1 provides mean values and standard deviations of these metrics, and p-values resulting from a two-tailed unequal variance t-test comparison of the means for each metric.

Metric 1: Safety: Attention

This is the total time of driver gaze that was related to driving, i.e., on the windshield or one of the three mirrors or out of the left/right window. More gaze time on driving suggests better attention to driving. However, as we can see from Table 6.4.1, there is not a significant difference in gaze time of the driver related to driving between the two conditions ($p>0.05$). We conclude that the total driver gaze time related to driving is unaffected by safe mode.

Metric 2: Safety: Time on Safety-Critical Goal

This is the total time needed for safety-critical goal accomplishment; lesser the time required, more efficient and safer the driving. As the mean time to accomplish the safety-critical goal of avoiding the vehicle that cuts in front is not significantly different ($p>0.05$) between the two conditions (safe mode on and off), we can conclude that Metric 2 is not useful because this metric is unaffected by safe mode.

Metric 3: Safety: Time on Driving Actions

This is the total time needed for the accomplishment of all (not just safety-critical) driving related actions. Ideally this should be same in both modes. And this is true here as all driving related actions take exactly the same time to finish their execution in either of the two conditions. Hence, this metric is unaffected by the safe mode.

Metric 4: Safety: Reaction Time for Safety-critical Actions

This is the delay between when any emergency driving-related action is created and it actually starts executing. This metric captures the response delay or the reaction time of the simulated driver. We can see from the t-test comparing the mean values of driver reaction time with the safe mode on and off that when the safe mode in on, there is a significant reduction

($p<0.05$) in the driver reaction time for safety-critical actions. Hence, safe mode is better because it facilitates a faster response by the driver.

Metric 5: Safety & Efficiency: Ratio of Safety-critical Reaction Time and Telematics Delay for Actions

This is the ratio of the delay between when any safety-critical action is created and when the driver starts executing it and the delay between when any telematics device related action is created and when the driver starts executing it. Ideally this should be less than 1 and small because we want the driver to be more responsive to safety-critical driving actions than actions on a telematics device such as a navigation device. When safe mode is on, the mean value of this ratio is significantly larger ($p<0.05$) than when safe mode is off.

However, this is not an indication that safe mode is actually unsafe. The reason for this is that with safe mode on, telematics actions are cancelled. Hence, the denominator of this ratio becomes very small, thereby increasing the ratio. In other words, safe mode does not actually increase the reaction time of the driver; as metric 4 showed, it in fact improves driver response (reduces reaction times for driving actions). Still, the ratio is larger when safe mode is on because it cancels telematics actions. So this particular metric is not useful for evaluating the effectiveness of the safe mode capability.

Metric 6: Safety: Interface Actions to Driving Actions Ratio

This is the ratio of the total number of telematics device related actions to the total number of driving related actions. Ideally, this should be less than 1 and small. In case of safe mode driving, the total number of driving actions is larger compared to the driver's telematics interface related actions; therefore, the ratio of the means of interface actions to driving actions is significantly smaller ($p<0.05$) in safe mode. Hence, keeping safe mode on improves driver's safety by facilitating more driving-related actions than interface-related actions.

Metric 7: Safety: Visual Perception Overload Extent

This is the maximum load of the visual channel above the baseline value of 100%. Table 6.4.1 shows that between the two conditions, there is not a significant difference in the extent of how much the driver's visual perception is overloaded ($p>0.05$). We conclude that visual perception overload extent is unaffected by safe mode.

Metric 8: Safety: Visual Perception Overload Time

This is the total time for which the driver's visual channel load is greater than or equal to 100%. Table 6.4.1 shows that between the two conditions, there is not a significant difference in the time for which the driver's visual perception is overloaded ($p>0.05$). We conclude that visual perception overload time is unaffected by safe mode.

Metric 9: Safety: Cognitive Processing Overload Extent

This is the maximum load of the cognition channel above the baseline value of 128%. The extent of cognitive overload is higher by 22% with safe mode off than with it on. Therefore, safe mode on is superior to safe mode off in terms of safety because it caused less of an overload.

Metric 10: Safety: Cognitive Processing Overload Time

This is the total time for which the driver's cognitive channel load is greater than or equal to the baseline value of 128%. Table 6.4.1 shows that between the two conditions, there is not a significant difference in the time for which the driver's cognitive channel is overloaded ($p>0.05$).

Though the time for which the driver's visual channel was overloaded was not different between the two conditions, as Metric 9 showed, safe mode on is still better with respect to safety.

6.5 Conclusion

From the results reported above, we can conclude that for the specific scenario simulated, keeping safe mode on while performing safety-critical maneuvering is better for driver’s safety; the driver reacts quicker and successfully accomplishes the safety-critical driving maneuver with less cognitive workload when the safe mode is on.

Chapter 7

Experiment: Static Delay vs. No Static Delay

In this experiment, we evaluated the effectiveness of the static delay capability of the Intelligent Driver Assistance subsystem. We used the events "right lane closed ahead" and "driver makes cell phone call" for this purpose. The idea behind this experiment was to find out if a futuristic IDA system is able to delay the operation of a telematics device by a fixed amount of time (e.g., suspend the ringing of a cell phone and later inform the driver and automatically dial the calling number) while the driver is engaged in an emergency driving maneuver, what effect will such a system have on driving safety?

7.1 Operation of Dynamic Delay

The static delay capability can be simulated by the CCDM for the following events:

1) Cell phone rings.

2) Driver enters a destination address on a navigation device by touch.

3) Driver enters a destination address on a navigation device by voice.

4) Navigation device issues a voice command.

5) Driver makes a cell phone call.

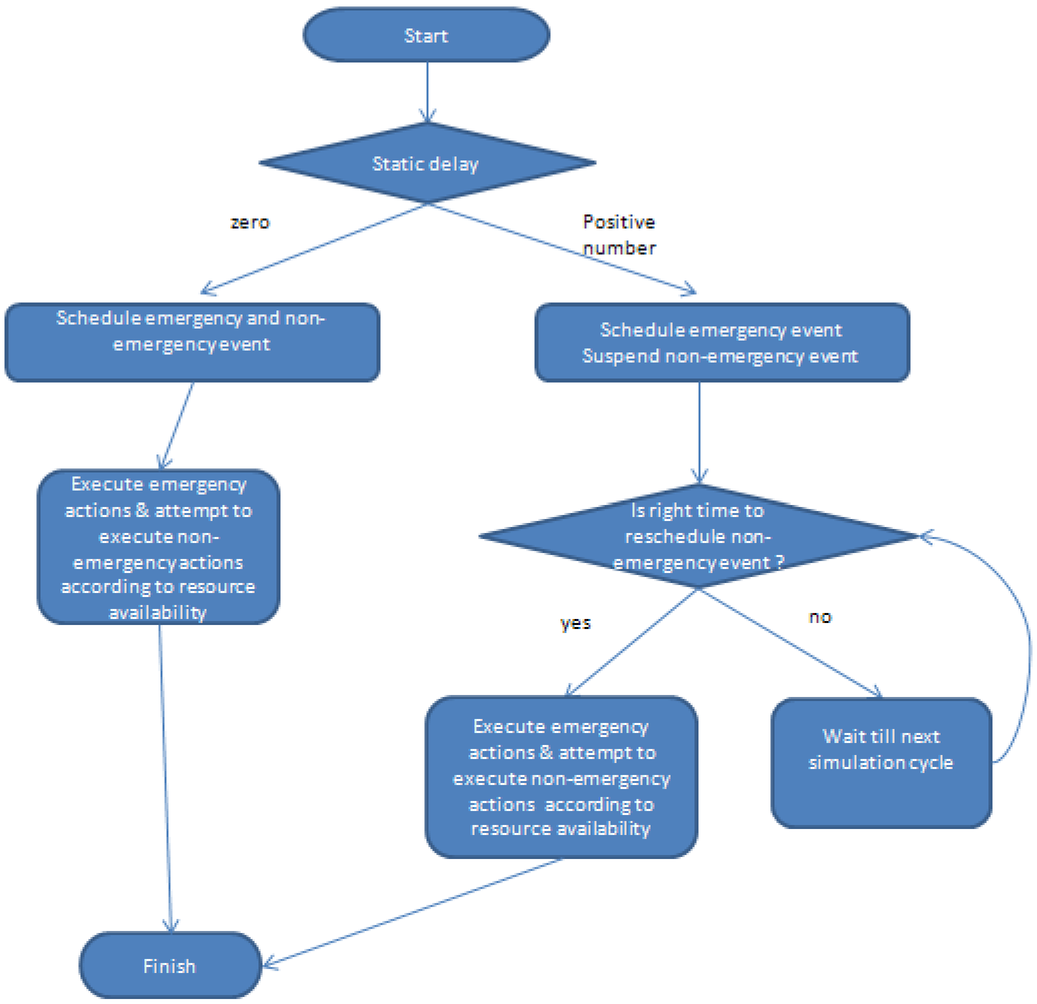

Start
Static delay
zero
Positive number
Schedule emergency and non-emergency event
Schedule emergency event Suspend non-emergency event
Execute emergency actions & attempt to execute non-emergency actions according to resource availability
Is right time to reschedule non-emergency event ?
yes
no
Execute emergency actions & attempt to execute non-emergency actions according to resource availability
Wait till next simulation cycle
Finish

Figure 7.1.1 Flowchart of Static Delay Mode

This capability is implemented in the CCDM as shown in Figure 7.1.1. The system interface allows the experimenter to specify a fixed time interval as the delay for each of the above events. If the driver is engaged in a safety-critical driving maneuver at the time when any of these events is scheduled to occur during the simulation, the IDA will delay the execution of the event till the specified delay time has elapsed. When the delay is over, the IDA will try to reschedule this event and the corresponding driver actions will be executed as long as sufficient resources are available for their execution.

Thus, this simulates an in-vehicle IDA that is able to detect, from driver and vehicle data, that a safety-critical driving maneuver is in progress, and is capable of temporarily delaying telematics device events (such as a cell phone ringing or the voice command from a navigation device) or temporarily disabling such a device to prevent the driver from operating it.

The intelligent agent can face two possible situations when it attempts to reschedule the non-emergency event after the pre-specified delay: 1) emergency actions are still executing, or 2) emergency maneuvers have finished and resources needed by the corresponding actions have been released. In the former situation, some actions related to the telematics event can face resource contention as the safety-critical actions are still executing. This may further delay the execution of the telematics event. In the latter situation, all actions related to the telematics event can be executed as sufficient resources are available.

7.2 Design of the Simulation Experiment

We selected the simulation parameters shown in table 7.2.1 for this experiment. The events along with their parameters in the simulated scenario for this experiment are shown in table 7.2.2 and table 7.2.3. In order to evaluate the effectiveness of the static delay mode, we ran the simulated driving scenario 10 times with static delay on and 10 times without static delay. Figure 7.2.1 shows the interface for specifying static delays.

Simulation		Unit
Simulation Length:	30000	ms
Driver Age:	Old	
Time of Day:	Night	
Weather:	Foggy	

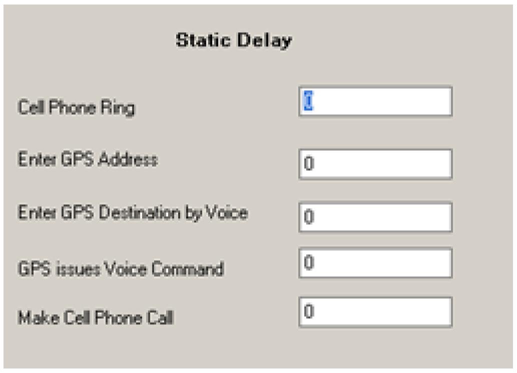
Table 7.2.1 Simulation Parameters

Event make cell phone call		Unit
Time:	3000	ms
Duration:	12000	ms
Probability:	100	
Priority:	3	

Table 7.2.2 Event 1 Parameters

Event right lane closed ahead		Unit
Time :	5000	ms
Distance:	120	m
Probability:	100	
Priority:	4	

Table 7.2.3 Event 2 Parameters



Static Delay

Cell Phone Ring	1
Enter GPS Address	0
Enter GPS Destination by Voice	0
GPS issues Voice Command	0
Make Cell Phone Call	0

Figure 7.2.1 Possible Events with Static Delay Function

7.3 Output Data

The output files generated by the simulation runs are the same as those described in Chapter 4.

7.4 Results

We analyzed the data in terms of the following 10 metrics to compare how the driver and driving are affected when the driver is dealing with the two events with the static delay mode on and with the static delay mode off. Table 7.4.1 provides mean values and standard deviations of these metrics, and p-values resulting from a two-tailed unequal variance t-test comparison of the means for each metric.

	Metric 1 (ms)	Metric 2 (ms)	Metric 3 (ms)	Metric 4 (ms)	Metric 5 (ratio)	Metric 6 (ratio)	Metric 7 (%)	Metric 8 (sec)	Metric 9 (%)	Metric 10 (sec)
static delay off mean (SD)	23276.5 (444.516)	13366 (102.518)	29050 (0)	23113 (320.98113)	4.150939 (0.0159337)	0.233611 (0.010958)	285 (0)	24.2 (0.6324)	244 (0)	24.2 (0.6324555)
static delay on mean (SD)	26752 (858.588)	9405.5 (850.073)	29050 (0)	3384 (2139.6907)	0.571198 (0.3517074)	0.202482 (0.01470)	200 (0)	28 (1.33333)	226 (0)	27.8 (1.4757295)
t-test p-value	0.0000000271	0.000000103	NA	0.000000000169646	0.000000000124582	0.000055135	NA	0.00000199	NA	0.000011565

Table 7.4.1 Experimental Data

Metric 1: Safety: Attention

This is the total time of driver gaze that was related to driving, i.e., on the windshield or one of the three mirrors or out of the left/right window. More gaze time on driving tasks suggests better attention to driving. Here, the mean time of driving related driver gaze when the static delay was off was significantly less ($p<0.05$) compared to when there was a static delay; hence, with respect to driver’s attention, static delay on is better than static delay off.

Metric 2: Safety: Time on Safety-Critical Goal

This is the total time needed for safety-critical goal accomplishment; lesser the time required, more efficient and safer the driving. As the mean time to finish the driver goal "avoid stationary obstacle" was significantly less ($p<0.05$) with the static delay on, we can conclude that driver’s safety is improved by static delay because the safety-critical driving goal is accomplished quicker.

Metric 3: Safety: Time on Driving Actions

This is the total time needed for the accomplishment of all (not just safety-critical) driving related actions. Ideally this should be same in both modes. And this is true here as all driving related actions takes exactly same time to finish their execution in either of the modes. Hence, we conclude that the total time of driving is unaffected by dynamic delay.

Metric 4: Safety: Reaction Time for Safety-critical Actions

This is the delay between when any emergency driving-related action is created and when it actually starts executing. This metric captures the response delay or reaction time of the simulated driver. We can see from the t-test ($p<0.05$) that when the static delay mode was on, there was a significant reduction in driver reaction times for safety-critical actions. Hence, the static delay mode helps the driver react more quickly in driving and thus improves safety.

Metric 5: Safety & Efficiency: Ratio of Safety-critical Reaction Time and Telematics Delay for Actions

This is the ratio of the delay between when any safety-critical action is created and when the driver starts executing it and the delay between when any telematics device related action is created and when the driver starts executing it. Ideally this should be less than 1 and small because we want the driver to be more responsive to safety-critical driving actions than actions on a telematics device such as a navigation device. When the static delay was on, the mean ratio was significantly smaller ($p<0.05$) than when static delay was off. The reason for this is that with static delay on, non-driving actions (i.e., telematics interface actions) are delayed by at least the pre-specified amount of time. Hence, the denominator of ratio is large compared to the numerator. Thus, this IDA mode inherently improves this ratio. Clearly, the static delay mode is better with respect to safety & efficiency.

Metric 6: Safety: Interface Actions to Driving Action Ratio

This is the ratio of the total number of telematics-device related actions to the total number of driving related actions. Ideally, this should be less than 1 and small. When static delay was on, the driver was able to do more driving-related actions. Therefore, the ratio of the mean number of interface actions to driving actions is significantly smaller ($p<0.05$) in the static delay on mode. Hence, adding a fixed amount of delay to interface related events when the driver is engaged in safety-critical driving maneuvers improves driver's safety by allowing the driver to focus more on driving actions.

Metric 7: Safety: Visual Perception Overload Extent

This is the maximum load of the visual channel above the baseline value of 100%. With static delay on, the extent of visual perception overload extent is lower compared to when the static delay mode is off. Hence, keeping the static delay on reduces the workload on the driver's visual perception and improves his safety.

Metric 8: Safety: Visual Perception Overload Time

This is the total time for which the driver's visual channel load is greater than or equal to 100%. Although the mean time for which the driver's visual perception load is over 100% when static delay is on is significantly larger (by 3.8 seconds) than with no static delay, since the extent of this overload is significantly smaller (by 85%) when compared to no static delay, we may still conclude that overall, static delay being on is better.

Metric 9: Safety: Cognitive Processing Overload Extent

This is the maximum load of the driver's cognition channel above the baseline value of 128%. The extent of cognitive overload is higher with no static delay mode than with the static delay mode on. Hence, it is concluded that the static delay mode reduces cognitive processing overload extent and thus improves driving safety.

Metric 10: Safety: Cognitive Processing Overload Time

This is the total time for which the driver's cognitive channel load is greater than or equal to the baseline value of 128%. Although the mean time for which the cognitive channel load is beyond 128% when static delay is on is significantly larger (by 3.6 seconds) than with no static delay, since the extent of this overload is significantly smaller (by 18%) when compared to no static delay, we may still conclude that overall, static delay on mode is better.

7.5 Conclusion

From the evaluation of all 10 metrics, it is clear that static delay on is superior to no static delay.

Chapter 8

Conclusion and Future Research

In this thesis we proposed, simulated, and analyzed four different modes of IDA to improve driving safety and efficiency during emergency maneuvering, while the driver is performing a secondary task such as entering an address into a GPS navigation device. After performing data analysis on 10 metrics of each mode of IDA, we concluded that keeping the safe mode on while performing safety-critical maneuvering was better for driver's safety. Also, the safety-critical task was completed in a lesser amount of time with the safe mode on, when compared to when the safe mode was off. The metrics did not provide sufficient evidence for the effectiveness of the dynamic delay mode as far as the driver's safety and efficiency were concerned. Overall, the extent of cognitive and visual perception overload on the driver and time for which this overload lasted were reduced in the limited cognitive resources mode. Static delay mode was superior to no static delay on 7 of the 10 metrics for safety and efficiency.

Limitations of this research include the following. Although we were able to find evidence for the effectiveness of two of the four proposed IDA modes, it should be noted that this evidence was based on data from simulation scenarios with specific emergency driving maneuvers and specific interface events. In order to be able to claim that our results generalize to all such event pairs, we will need to conduct several more experiments covering all possible event pairs with different parameters. Furthermore, human experimental data will need to be collected using driving simulators and working prototypes of IDA, and compared with simulated data, in order validate and generalize our current findings. Thus, the applicability of our findings is at present only within the context of the experiments and the CCDM.

In future research, we will improve the performance of the safe mode of the IDA based on the following considerations:

Suppose the following is the state of the simulated driver at an instant, with two driver goals active and a third emergency maneuver goal scheduled, when the safe mode IDA is on.

Goal Answer Cell Phone Call Priority = 3	Goal Enter Address on Navigation Device Priority = 2	Goal Avoid Moving Vehicle In front Priority = 4
Executing	Executing	Scheduled

The IDA will terminate both cell phone and navigation device goals (thereby simulating the disablement of these devices) because it detects that the driver needs to engage in an emergency maneuver.

Goal Avoid Moving Vehicle In front Priority = 4	~~Goal Answer Cell Phone Call Priority = 3~~	~~Goal Enter Address on Navigation Device Priority = 2~~
Executing	Terminated	Terminated

In the current version of the Cognitive-Computational Driving Model, when the safe mode is on, IDA will simply terminate all goals and corresponding actions with priority less than 4 if a priority 4 event or goal (indicating an emergency maneuver) is scheduled. The priority 4 goal will take control of the goal queue and starts executing. This is a greedy behavior, because there is always the possibility that the priority 4 goal may not consume all available resources, and so the driver may in fact be able to engage in actions corresponding to one or more of other lower priority goals (such as talking on the phone).A similar situation is the following:

Goal Avoid Moving Vehicle In front Priority = 4	Goal Answer Cell Phone Call Priority = 3	Goal Enter Address on Navigation Device Priority = 2
Scheduled	Scheduled	Scheduled

The IDA will delay the execution of goals with priority less than 4:

Goal Avoid Moving Vehicle In front Priority = 4	Goal Answer Cell Phone Call Priority = 3	Goal Enter Address on Navigation Device Priority = 2
Executing	Delayed	Delayed

If an emergency goal gets scheduled prior to non-emergency goals and starts executing, all non-emergency goals are suspended by the IDA until the emergency goal either fails or finishes successfully. In this case again, there may actually be enough resources available to execute some actions of scheduled non-emergency goals.

To address the above-mentioned problems, we propose the following improvements to the safe mode implementation in the CCDM architecture.

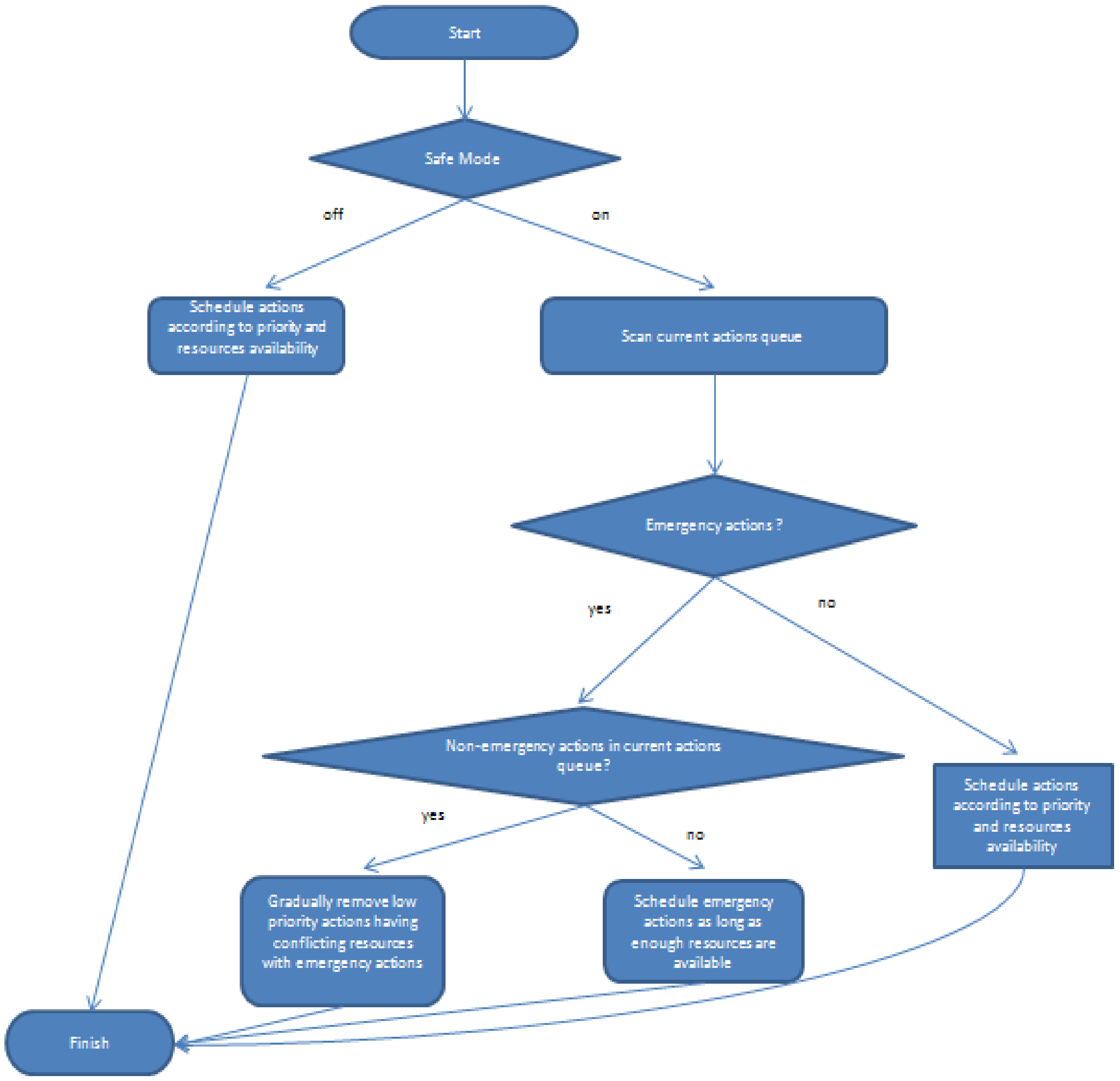

Start
Safe Mode
off
on
Schedule actions according to priority and resources availability
Scan current actions queue
Emergency actions ?
yes
no
Non-emergency actions in current actions queue?
Schedule actions according to priority and resources availability
yes
no
Gradually remove low priority actions having conflicting resources with emergency actions
Schedule emergency actions as long as enough resources are available
Finish

Figure 8.1 Modification1 to the Safe Mode

The proposed modification to the safe mode IDA in the above figure will gradually remove low priority actions which have conflicting resources with scheduled emergency actions. Hence, not all lower priority actions will be terminated as a result of the scheduling of emergency actions; some of them can still finish executing along with emergency actions. This way, safe mode IDA will act as a true priority based scheduler of actions by the driver.

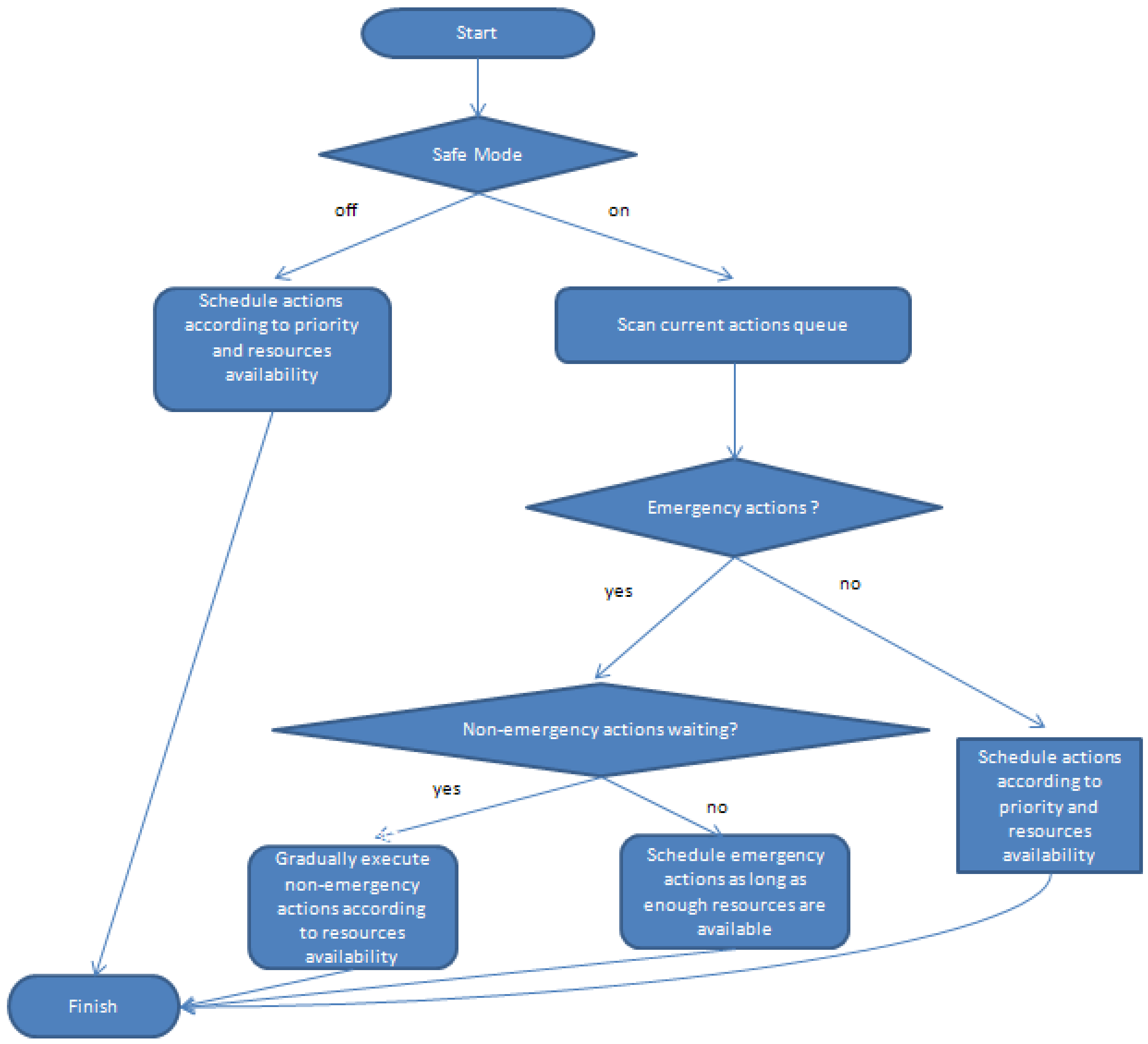

Start
Safe Mode
off
on
Schedule actions according to priority and resources availability
Scan current actions queue
Emergency actions ?
yes
no
Non-emergency actions waiting?
Schedule actions according to priority and resources availability
yes
no
Gradually execute non-emergency actions according to resources availability
Schedule emergency actions as long as enough resources are available
Finish

Figure 8.2 Modification2 to the Safe Mode

The proposed modification to the safe mode in the above figure will allow non-emergency actions to be executed along with emergency actions, provided these don't have conflicting resource requirements. Hence, non-emergency actions do not necessarily have to wait until emergency actions finish their execution. We believe that these modifications will enhance the effectiveness of the safe mode of IDA.

Another problem with the current CCDM implementation is that its interface allows an event to take place only once during one simulation run. This prevents an experimenter from measuring the driver’s performance in scenarios with repeated events. If the interface of the CCDM is modified to give the user the ability to go back and forth on the input forms during scenario specification, this will give the user more freedom to review and change scenario specifications and corresponding parameter values. Also, with the scheduling of each event, a pop-up window should ask the user if there is a need for a number of repetitions of the same event and it should allow the user to schedule an event multiple times with same or different parameters in the driving scenario. With these modifications, the CCDM will become more flexible in terms of the driving scenarios that can be specified and simulated.

References

[1] Brumby, D.P., Salvucci, D.D., and Howes, A. An Empirical Investigation into Dual-Task Trade-offs while Driving and Dialing. in *Proceedings of the 21st BCS HCI Group Conference*, Swindon, UK, 2007, pp. 11-14.

[2] Zhang, X., Mei, T., Wang, R., and Mao, X. A Kind of Cognitive Driver Assistance System with Adaptive Authority Adjusting. http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=05164463, 2009.

[3] Wu, C., Tsimhoni, O., and Liu, Y. Development of an Adaptive Workload Management System using Queuing Network-Model of Human Processor. in *51st Annual Conference of the Human Factors and Ergonomics Society*, Baltimore, MD, USA, 2007.

[4] Tanaka, S. I., Kawagoe, H., and Kondo, S. Workload of Using a Driver Assistance System. in *Proceedings of the IEEE Conference on Intelligent Transportation Systems*, 2000, pp. 382-386.

[5] May, J. F., Baldwin, C. L., and Parasuraman, R. Prevention of Rear-End Crashes in Drivers with Task-Induced Fatigue through the Use of Auditory Collision Avoidance Warnings. in *Proceedings of Human Factors and Ergonomics Society 50th Annual Meeting,* San Francisco, CA, USA, 2006.

[6] Kanevsky, D., Churchill, B., Faisman, A., and Nahamoo, D. Safety Driver Manager. IBM Research Report, IBM Research Division Thomas J. Watson Research Center, Yorktown Heights, NY, USA, 2004.

[7] Arroyo, E., Sullivan, S., and Selker, T. CarCoach: A Polite and Effective Driving Coach. in *Proceedings of the CHI Conference on Human Factors in Computing Systems* ,Montreal, Canada, 2006, pp. 357-362.

[8] Lee, J. D., Hoffman, J. D., and Hayes, E. Collision Warning Design to Mitigate Driver Distraction. *Proceedings of the SIGCHI Conference on Human factors in Computing Systems,* Vienna, Austria, 2004, pp. 65-72.

[9] Sakaguchi, Y., Okuwa, M., Takiguchi, K., and Akamatsu, M. Measuring and Modeling of Driver for Detecting Unusual Behavior for Driving Assistance. in *Proceedings of 18th International Conference on Enhanced Safety of Vehicles*, 2003.

[10] Kozak, K., Pohl, J., Birk, W., Greenberg, J., Artz, B., Blommer, M., Cathey, L., and Curry, R. Evaluation of Lane Departure Warnings for Drowsy Drivers. in *50th Annual Proceedings of Human Factors and Ergonomics Society*, San Francisco, CA, USA, 2006, pp. 2400-2404.

[11] Donmez, B., Boyle, L. N., Lee, J. D., and McGehee, D. V. Drivers attitudes toward imperfect distraction mitigation strategies. in *Transportation Research Part F: Traffic Psychology and Behavior,* vol. 9, no. 6, pp. 387-398, 2006.

[12] Ho, C., Reed, N., and Spence, C. Multisensory In-Car Warning Signals for Collision Avoidance. *Journal of the Human Factors and Ergonomics Society,* vol. 49, no. 6, pp. 1107-1114, 2007.

[13] Keller, J. Human Performance Modeling for Discrete-Event Simulation: Workload. in *Proceedings of the 34th Conference on Winter Simulation: Exploring New Frontiers.* San Diego, CA, USA, 2002, pp. 157-162.

[14] Wood, C., and Hurwitz, J. Driver workload management during cell phone conversations. in *Proceedings of the Third International Driving Symposium on Human Factors in Driving Assessment, Training and Vehicle Design,* Rockport, ME, USA, 2005, pp. 203-210.

[15] Liu, A., and Salvucci, D. D. Modeling and Prediction of Human Driver Behavior. in *Proceedings of the Ninth International Conference on Human-Computer Interaction*. New Orleans, LA, USA, 2001, Lawrence Erlbaum Associates, pp. 1479-14839.

[16] Salvucci, D. D. Modeling Tools for Predicting Driver Distraction. in *Proceedings of Human Factors and Ergonomics Society 49th Annual Meeting,* Santa Monica, CA, USA, 2005.

[17] Yuji, T., Nobuyuki, K., and Osamu, S. Assessment of drivers' workload under driving assistance systems. in *Proceedings of JSAE Annual Congress*, vol. 20, no. 01, pp. 15-20, 2001.

[18] Bierbaum, C.R., Szabo, S. M., and Aldrich, T. B. A Comprehensive Task Analysis of the UH-60 Mission with Crew Workload Estimates and Preliminary Decision Rules for Developing a UH-60 Workload Prediction Model. Technical Report ASI690-302-87[B], vol. I, II, III, IV, Anacapa Sciences, Fort Rucker, AL, USA, 1987.

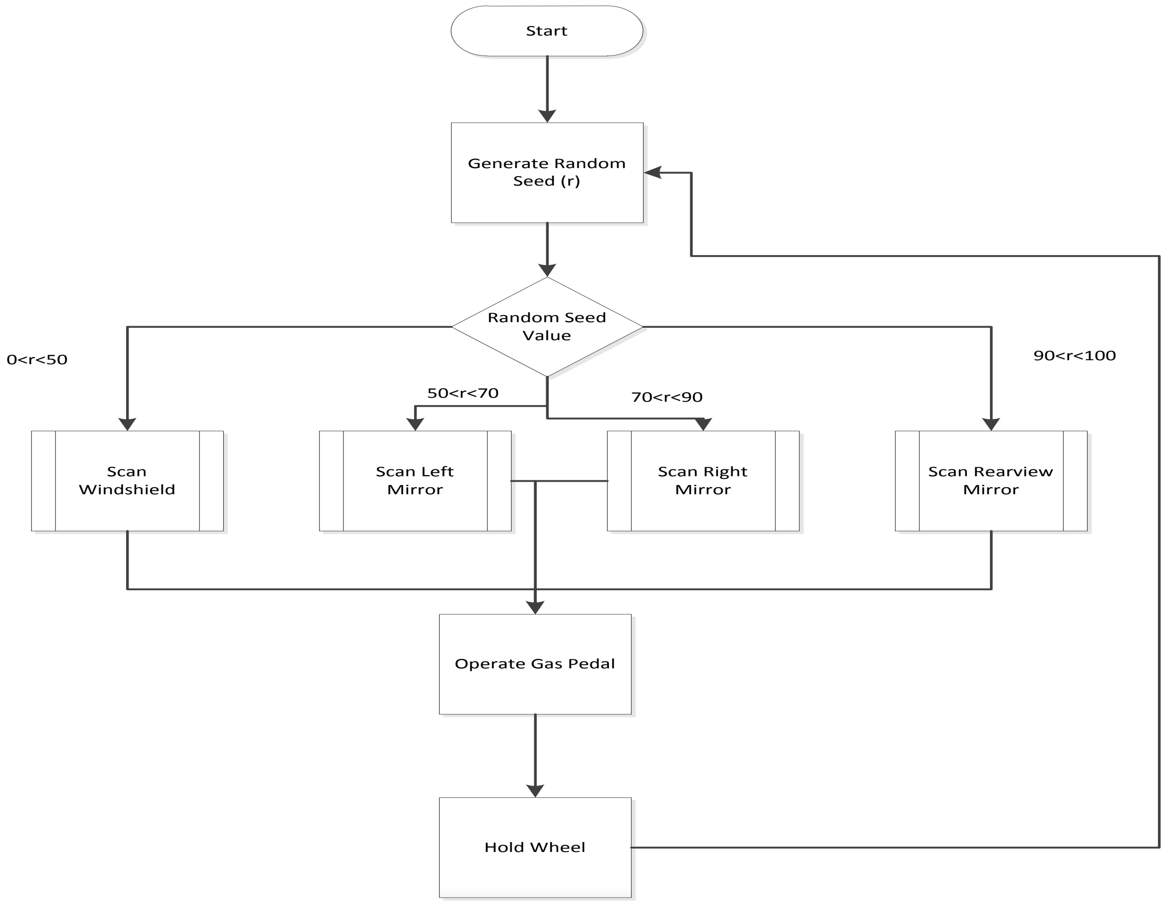
[19] Winter, J. C. F., Mulder, M., Paassen, M. M., Abbink, D. A., and Wieringa. P. A. A Two-Dimensional Weighting Function for a Driver Assistance System. *IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics,* vol. 38, no. 01, pp. 189-195, February 2008.

Appendix 1

Goal Specifications and Flowcharts

G-Drive (status: scheduled/executing/success/failure, priority: 1/2/3/4, scheduled-starting-time: GSS-clock, parent: goal instance or null)

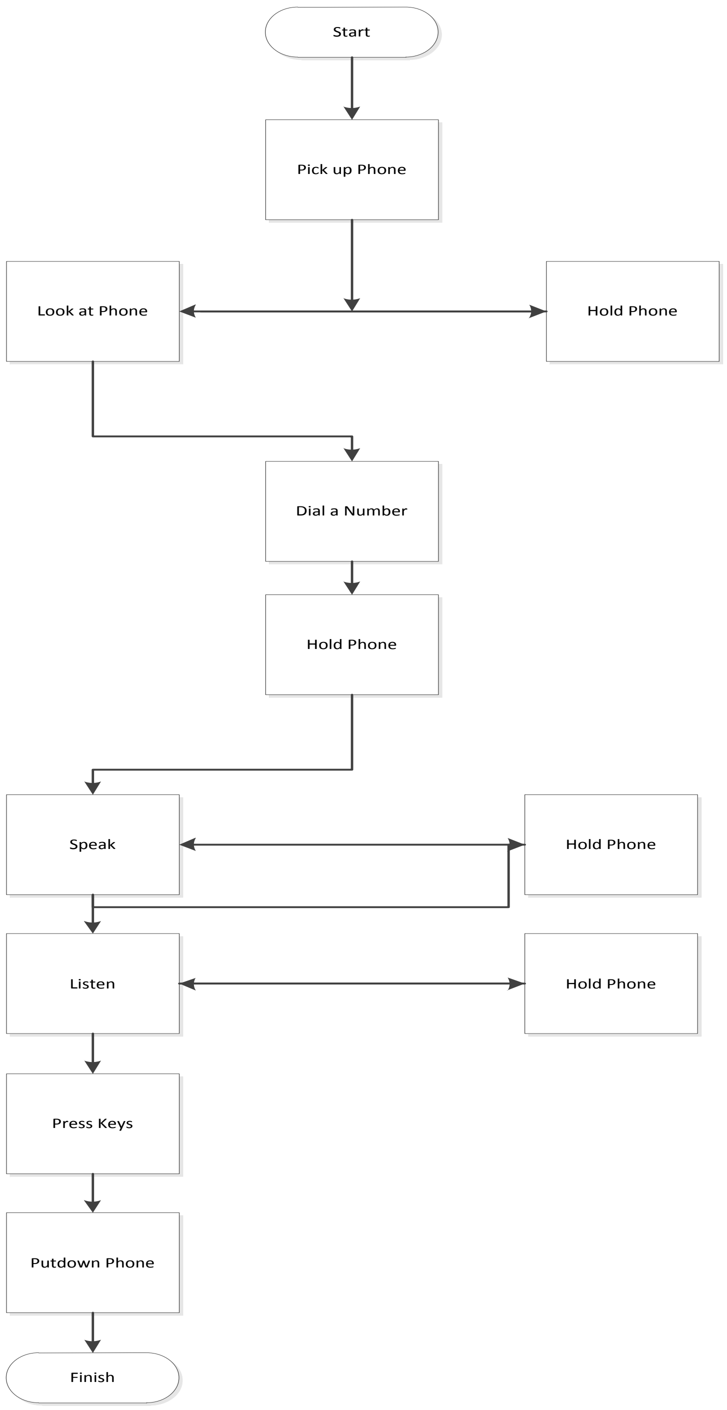
{Scan forward 50% of the time, left and right mirrors 20% of the time and rearview mirror 10% of the time. Hold the wheel and press steadily on the gas pedal. **Notes**: Simulation Engine should place this goal in Goal Q before simulation loops start with Actual Starting Time set to one delta-t. Since this is a goal that is constantly executing, it is suspended temporarily when other driving related goals start executing, and it is inserted again when such goals finish their execution. The goal is moved to the Goal History whenever interrupted and after the simulation ends.}



Flowchart of G-Drive

G-make-phone-call (duration: msec, status: scheduled/executing/success/failure, priority: 1/2/3/4, scheduled-starting-time: GSS-clock, parent: goal instance or null)

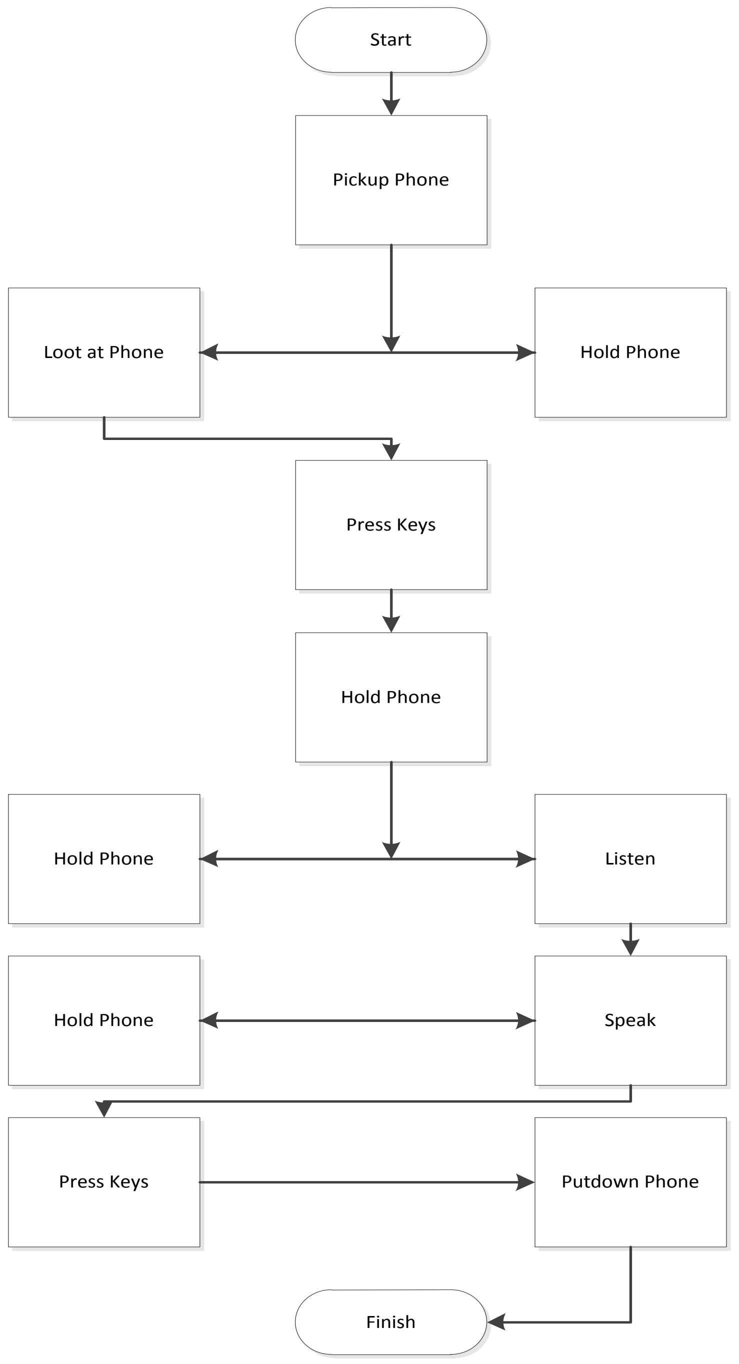
{Pick up the phone, look at it, dial 10 digits, hold the phone to ear and wait to get connected, start speaking, continue to alternately listen and speak while holding, end the call by pressing the end button, and put down the phone.}



Flowchart of G-make-phone-call

G-receive-phone-call (duration: msec, status: scheduled/executing/success/failure, priority: 1/2/3/4, scheduled-starting-time: GSS-clock, parent: goal instance or null)

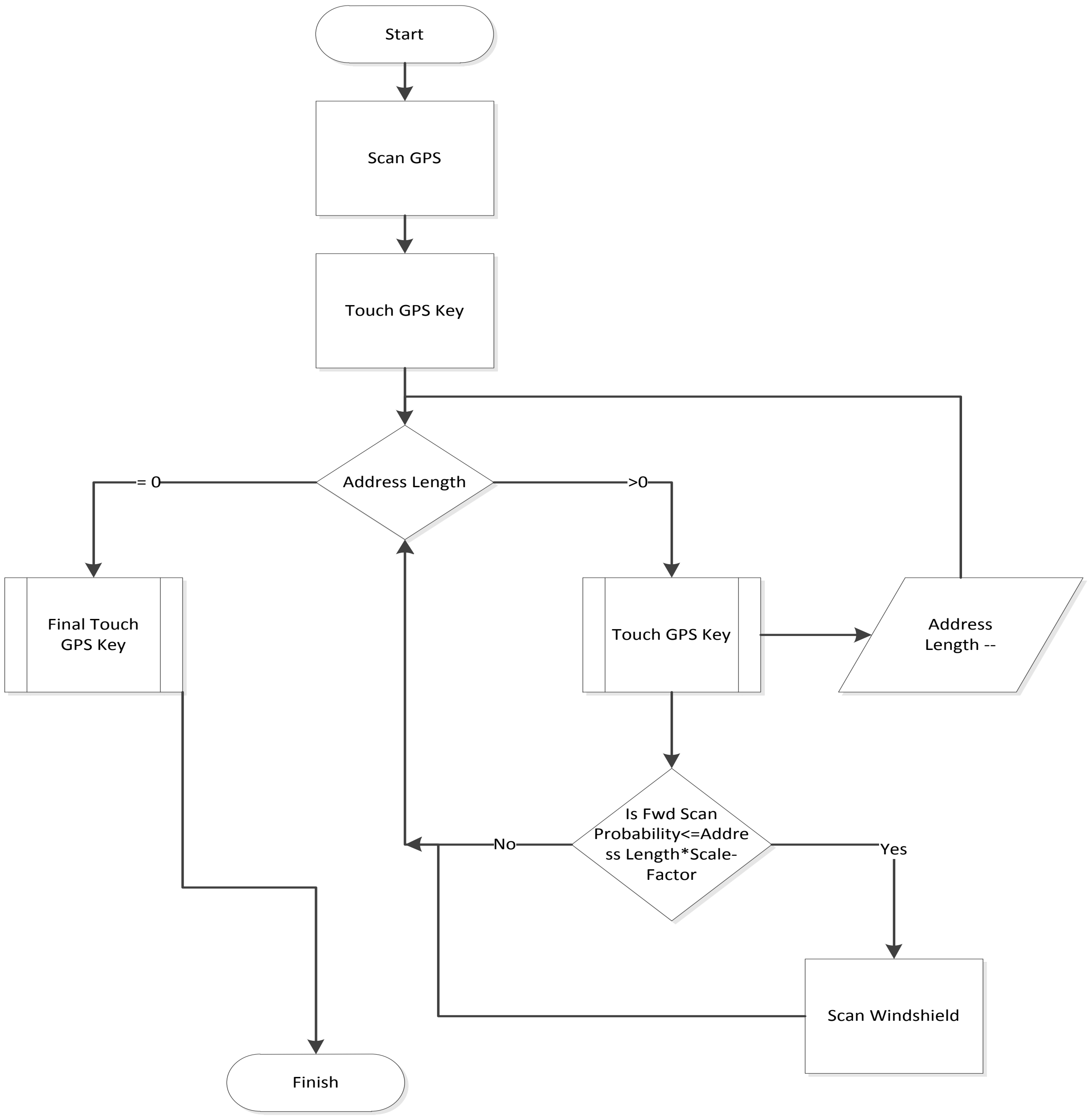
{Pick up the phone, look at it, press the answer button, hold the phone to ear and wait to get connected, start listening, continue to alternately speak and listen while holding, end the call by pressing the end button, and put down the phone.}



Flowchart of G-receive-phone-call

G-enter-GPS-address (status: scheduled/executing/success/failure, priority: 1/2/3/4, address-length: integer > 0, scheduled-starting-time: GSS-clock, parent: goal instance or null)

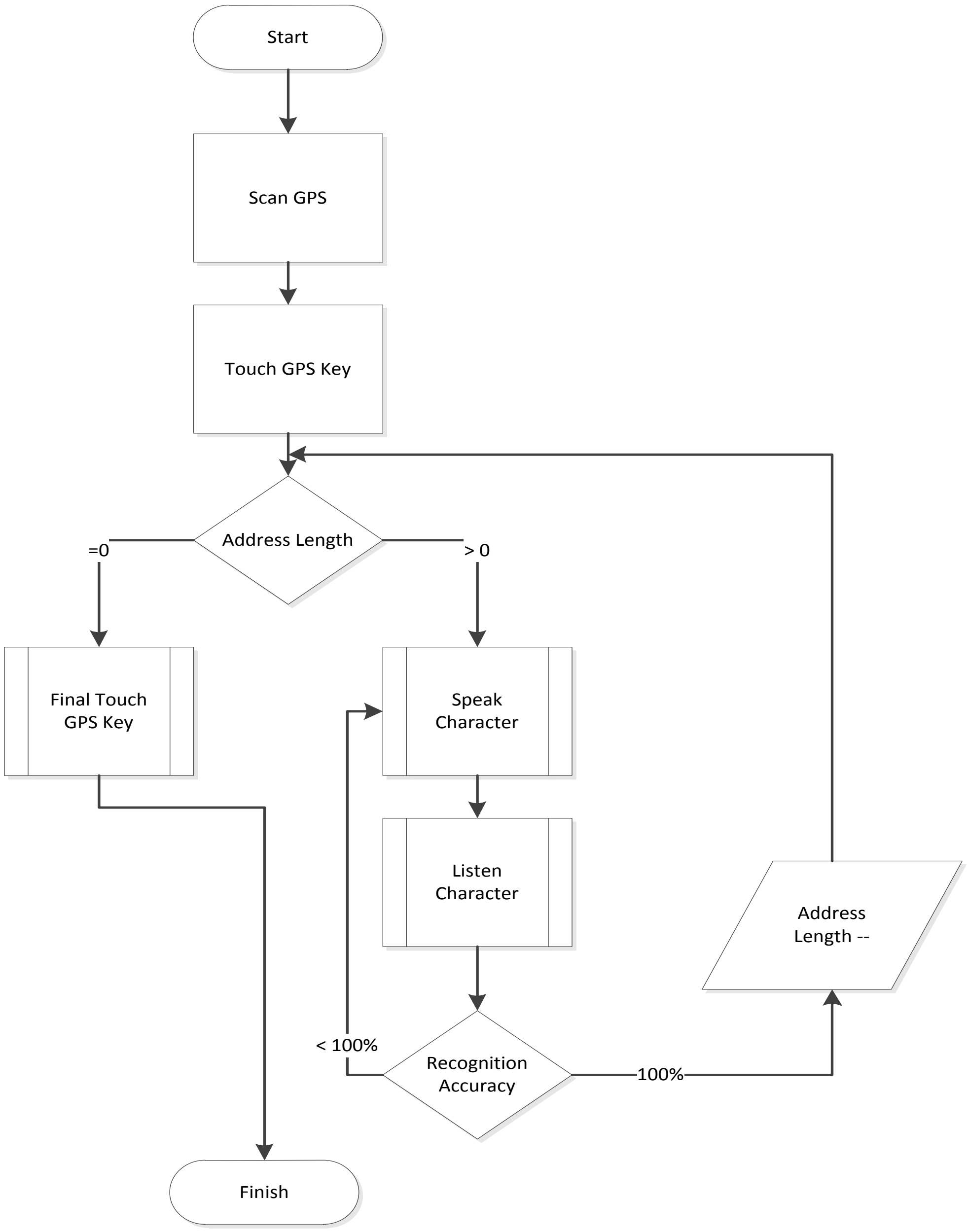
{Look at the navigation system, select the keypad mode with one button press while looking at it, enter the destination address one character at a time either by pressing a key on a keyboard of the GPS or by touching a key on the touch screen of the GPS while looking at the GPS, after each character entry scan forward with a probability proportional to length of destination address – i.e., the longer the address the more likely is the driver to look forward between typing, finally enter the address with one button press/touch while looking at the GPS.}



Flowchart of G-enter-GPS-address

G-GPS-enter-destination-by-spoken-character (status: scheduled/executing/success/failure, priority: 1/2/3/4, address-length: integer > 0, recognition-rate: 1-100, scheduled-starting-time: GSS-clock, parent: goal instance or null)

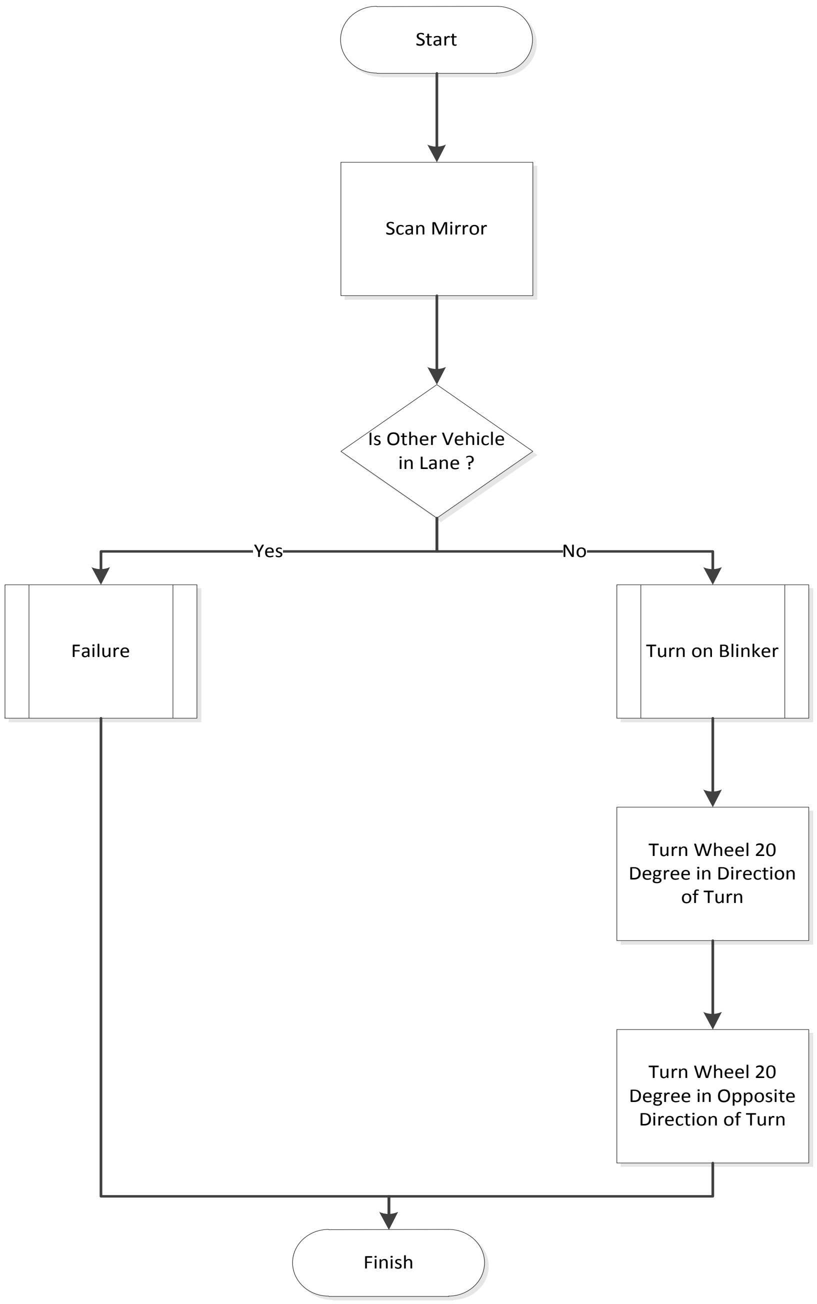
{Look at the navigation system, select the keypad mode with one button press while looking at it, enter the destination address one character at a time by speaking without looking at the GPS, listen to GPS repeating the character and correct any errors, finally enter the address with one button press/touch while looking at the GPS.}



Flowchart of G-GPS-enter-destination-by-spoken-character

G-lane-change (direction: left/right, status: scheduled/executing/success/failure, priority: 1/2/3/4, scheduled-starting-time: GSS-clock, parent: goal instance or null)

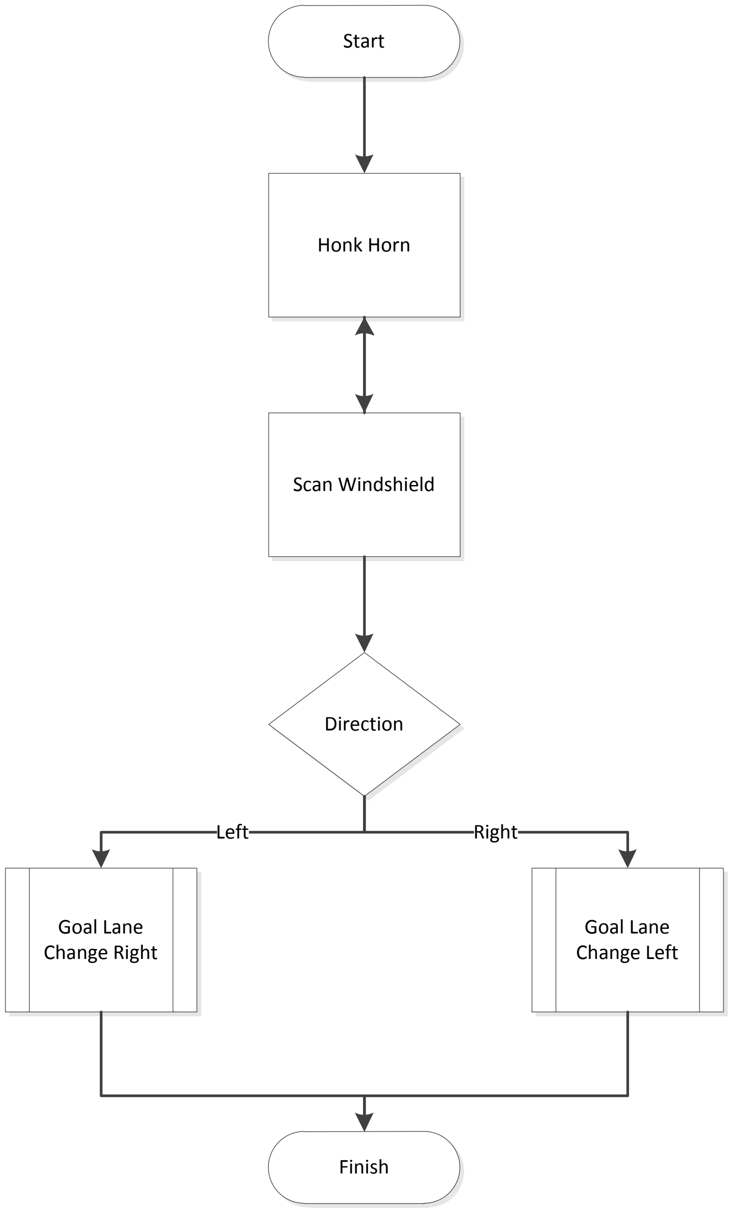
{Scan the lane to which lane change is needed; wait till the scan completes and succeeds. Then check the corresponding GSS variable to see if there is a vehicle in that lane; if so, lane change fails; else these three actions are to be executed one after the other in this order: operate blinker, turn steering wheel in the direction of lane change, and turn the wheel in the opposite direction. Then continue driving.}


Start
Scan Mirror
Is Other Vehicle in Lane ?
Yes
No
Failure
Turn on Blinker
Turn Wheel 20 Degree in Direction of Turn
Turn Wheel 20 Degree in Opposite Direction of Turn
Finish

Flowchart of G-lane-change

G-avoid-drifting-vehicle (status: scheduled/executing/success/failure, priority: 1/2/3/4, parent: goal instance or null, scheduled-starting-time: GSS-clock, direction: FromLeft/FromRight)

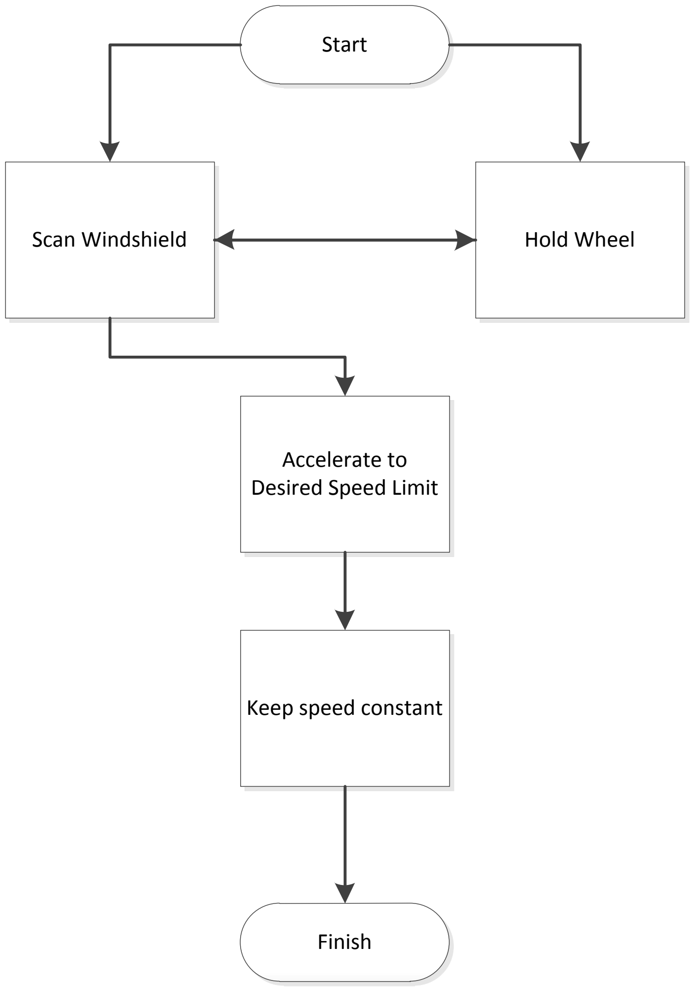
{When the vehicle is moving on the road, and another vehicle comes either from left or right direction, then change the lane appropriately. E.g., when the vehicle is drifting too close from left, then execute goal lane change to right and vice versa. If the goal lane change fails, accident occurs else goal succeeds when lane change finishes successfully.}



Flowchart of G-avoid-drifting-vehicle

G-resume-normal-drive (status: scheduled/executing/success/failure, priority: 1/2/3/4, speed: KMH, scheduled-starting-time: GSS-clock, parent: goal instance or null)

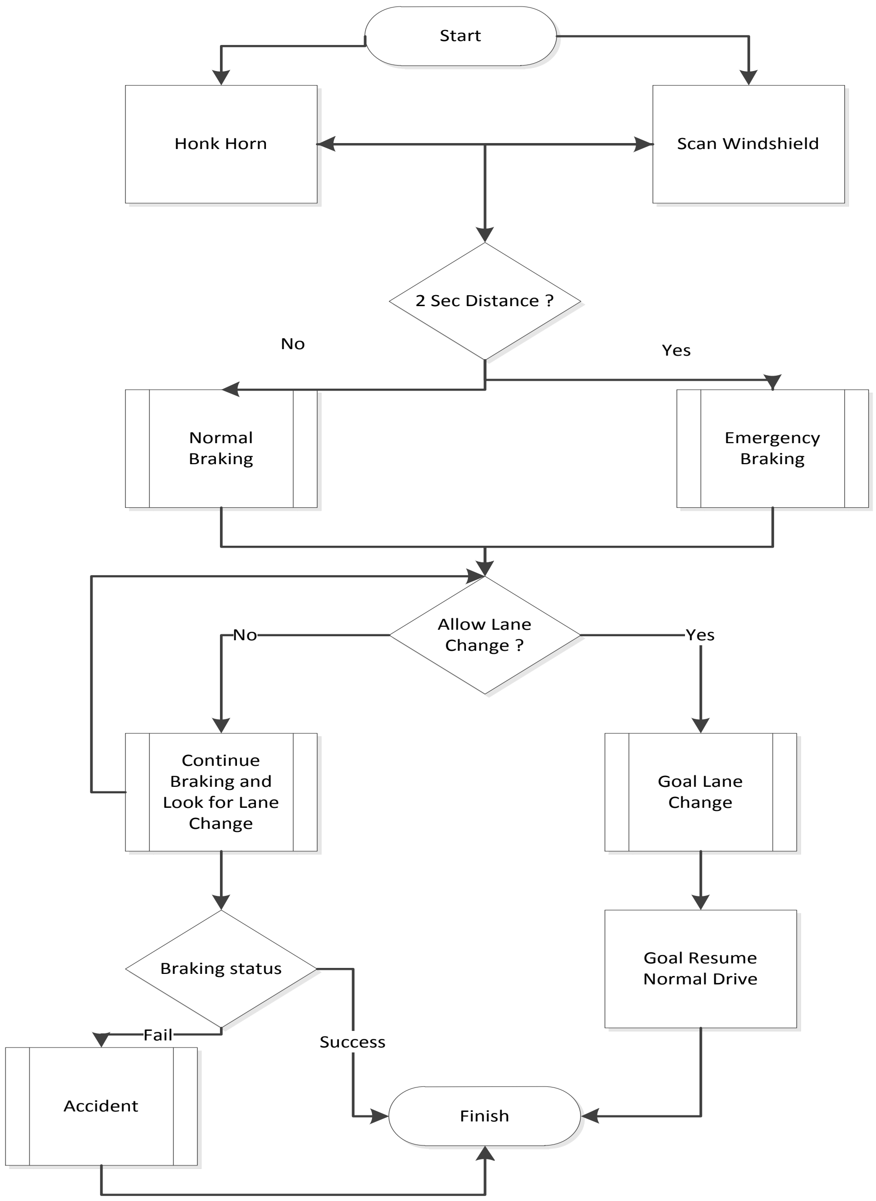
{Look forward and hold the steering wheel steady, and press steadily on the gas pedal to accelerate to speed limit. This goal represents the driver's intention to resume normal driving after a braking action.}



Flowchart of G-resume-normal-drive

G-avoid-stationary-obstacle-in-front (target-speed: 0 meters/millisecond, target-location: meters, status: scheduled/executing/success/failure, priority: 1/2/3/4, parent: goal instance or null, scheduled-starting-time: GSS-clock)

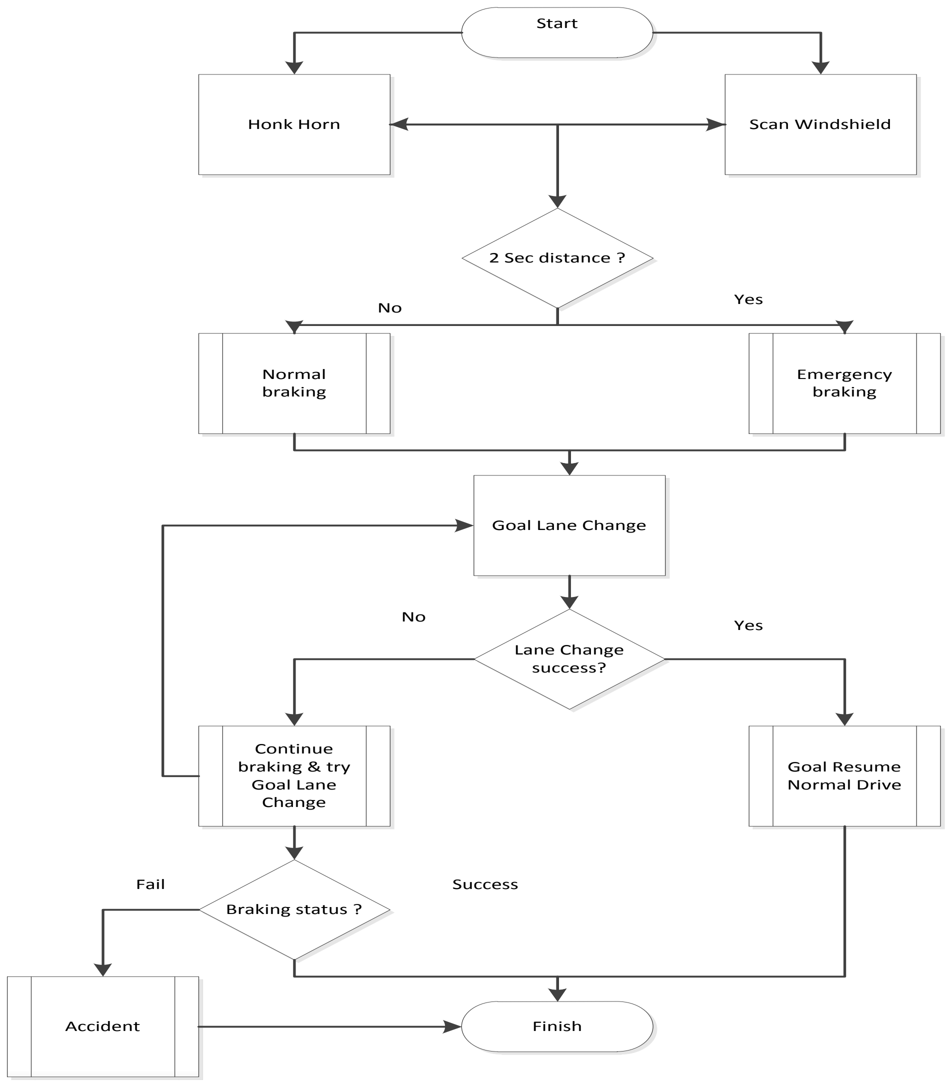
{Once a stationary obstacle in front (like a rock or a closed lane) is noticed, if at a safe distance from the obstacle, brake normally. If vehicle does not stop before reaching the obstacle then braking fails – i.e. an accident happens. If a multi-lane road, see if vehicle can change to left or right lane. If so, change lane, and resume normal driving after lane change succeeds. If a lane change is not possible, continue braking and continue looking for lane change opportunities until lane change succeeds or until vehicle comes to a stop; if lane change does not succeed and vehicle does not stop before reaching the obstacle then braking fails – i.e. an accident happens. }



Flowchart of G-avoid-stationary-obstacle-in-front

G-avoid-moving-vehicle-in-front (status: scheduled/executing/success/failure, priority: 1/2/3/4, parent: goal instance or null, scheduled-starting-time: GSS-clock)

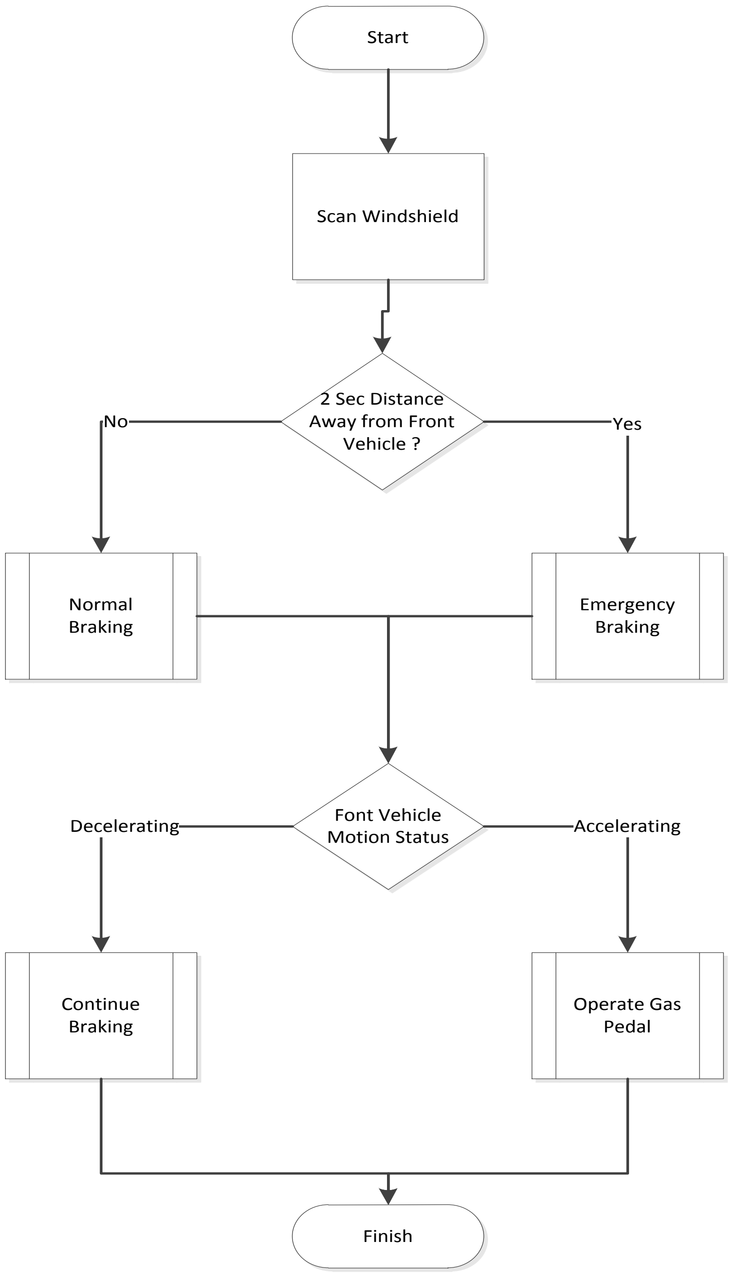
{Once a moving obstacle in front (like a suddenly slowing vehicle in front) is noticed, brake normally. If vehicle is unable to slow down quickly enough and reaches the obstacle then braking fails – i.e. an accident happens. If a multi-lane road, see if vehicle can change to left or right lane. If lane change succeeds, stop braking, and resume normal driving. If a lane change is not possible, continue braking and continue looking for lane change opportunities until lane change succeeds or until vehicle slows to the same speed as the moving obstacle in front; if lane change does not succeed and vehicle reaches the obstacle then braking fails – i.e. an accident happens. If the vehicle able to change lanes, it will proceed; if not, it will attempt to slow to the speed of the vehicle in front and maintain a safe (2 second) distance from it. If the vehicle is unable to change the lane, an accident will occur. This goal will keep executing as long as there is a front vehicle moving at less than the road speed limit or decelerating.}



Flowchart of G-avoid-moving-vehicle-in-front

G-safely-follow-vehicle-in-front (priority: 1/2/3/4, scheduled-starting-time: GSS-clock, parent: goal instance or null, status: scheduled/executing/success/failure)

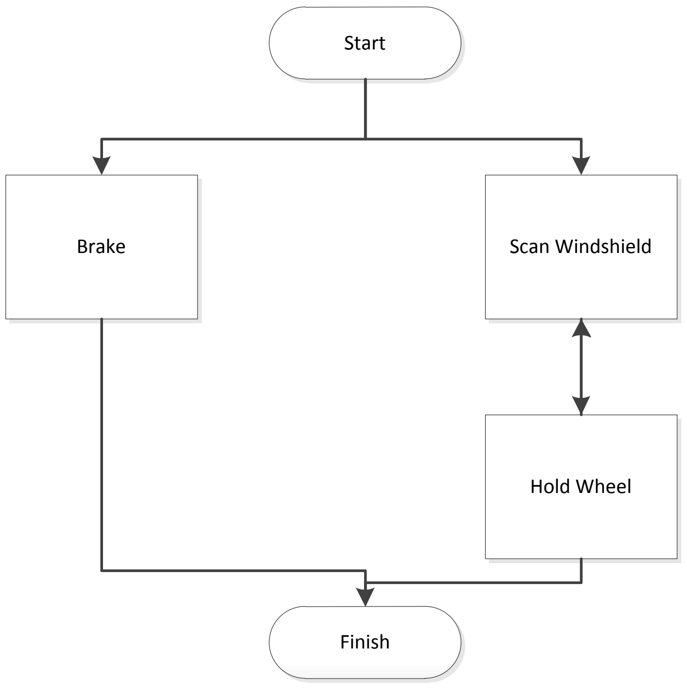
{Bring vehicle to at most the same speed as front vehicle and at least 2 seconds away from it. This goal is triggered by events that model the front vehicle moving at a speed less than the simulated vehicle, and accelerating or decelerating. The goal succeeds when the simulated vehicle reaches a speed less than or equal to the front vehicle's speed and is at least 2 seconds away from it. In this case braking is stopped and normal driving resumed.}



Flowchart of G-safely-follow-vehicle-in-front

G-stop (target-location: meters, priority: 1/2/3/4, parent: goal instance or null, scheduled-starting-time: GSS-clock, status: scheduled/executing/success/failure)

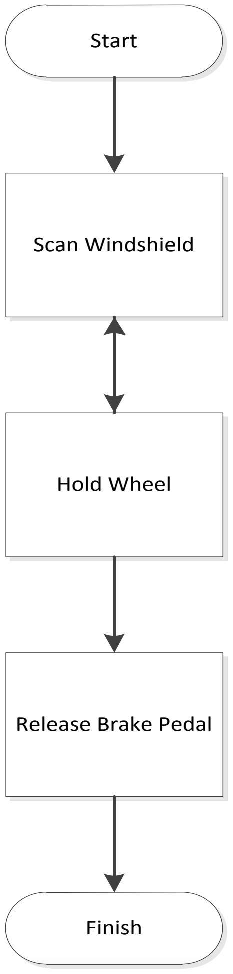
{Bring vehicle to a stop at or before reaching a fixed target-location}



Flowchart of G-stop

G-start-after-stopping (status: scheduled/executing/success/failure, priority: 1/2/3/4, scheduled-starting-time: GSS-clock, parent: goal instance or null)

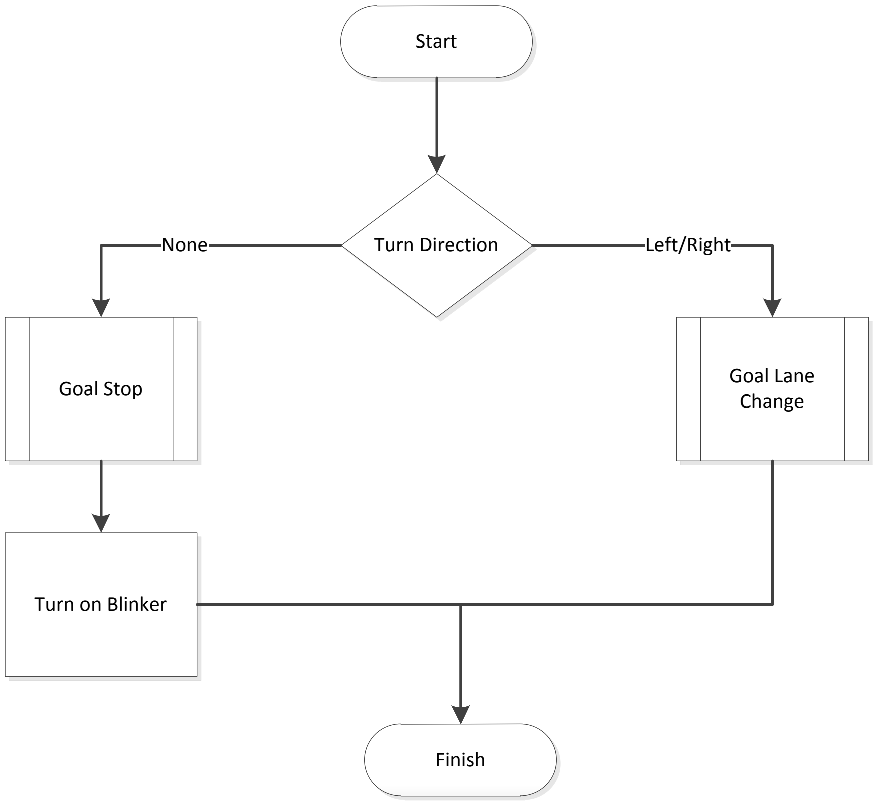
{Look forward, release brake pedal and hold the wheel steady. This goal represents the driver's intention to start the vehicle moving again after it stops but not to accelerate to the road's speed limit or to do normal driving.}



Flowchart of G-start-after-stopping

G-stop- before-turn (target-location: meters, turndirection: left/right, priority: 1/2/3/4, parent: goal instance or null, scheduled-starting-time: GSS-clock, status: scheduled/executing/success/failure)

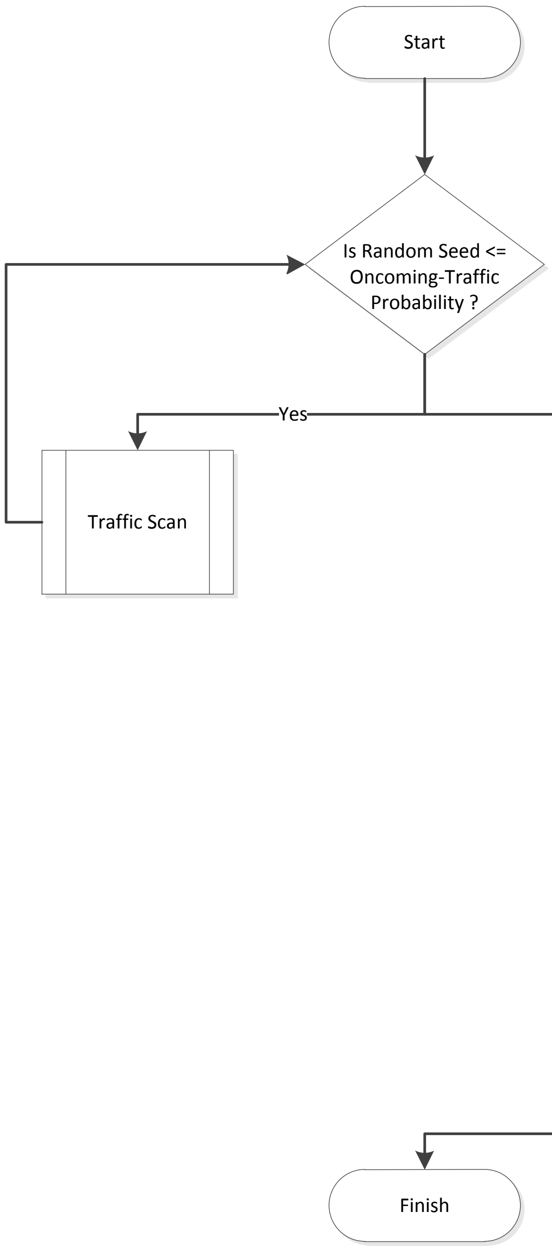
{Change the lane to make appropriate turn then stop the vehicle. If the vehicle fails to reach the desired lane by target location or reaches the correct lane but fails to come to a stop, it will continue moving as it can't make the turn.}



Flowchart of G-stop- before-turn

G-start-after-stopping-when-traffic-clears (oncoming-traffic-probability: 0-100, oncoming-traffic-direction: left/right, status: scheduled/executing/success/failure, priority: 1/2/3/4, scheduled-starting-time: GSS-clock, parent: goal instance or null)

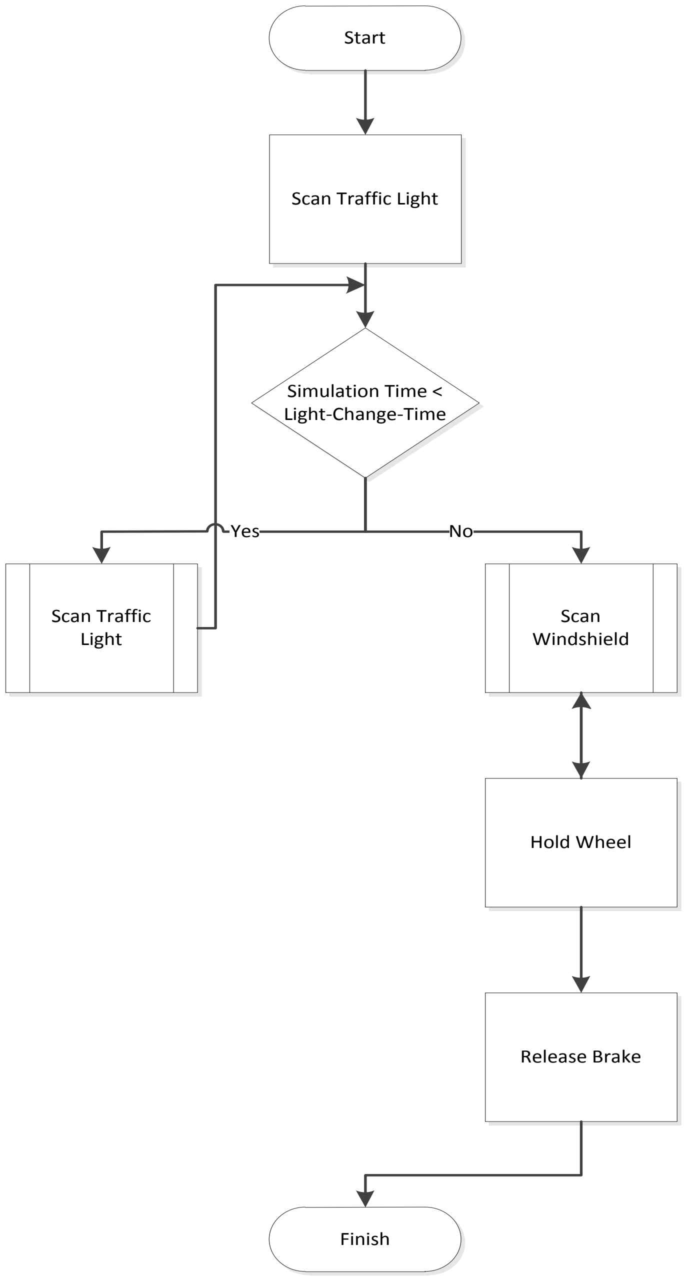
{Scan oncoming-traffic-direction until traffic is clear – modeled as when a random number falls outside the oncoming-traffic-probability; then look forward, release brake pedal and hold the wheel steady. This goal represents the driver's intention to start the vehicle moving again (but not to accelerate to normal speed) after it stops for oncoming traffic.}



Flowchart of G-start-after-stopping-when-traffic-clears

G-start-after-stopping-when-traffic-light-turns-green (traffic-light-duration: milliseconds, status: scheduled/executing/success/failure, priority: 1/2/3/4, scheduled-starting-time: GSS-clock, parent: goal instance or null)

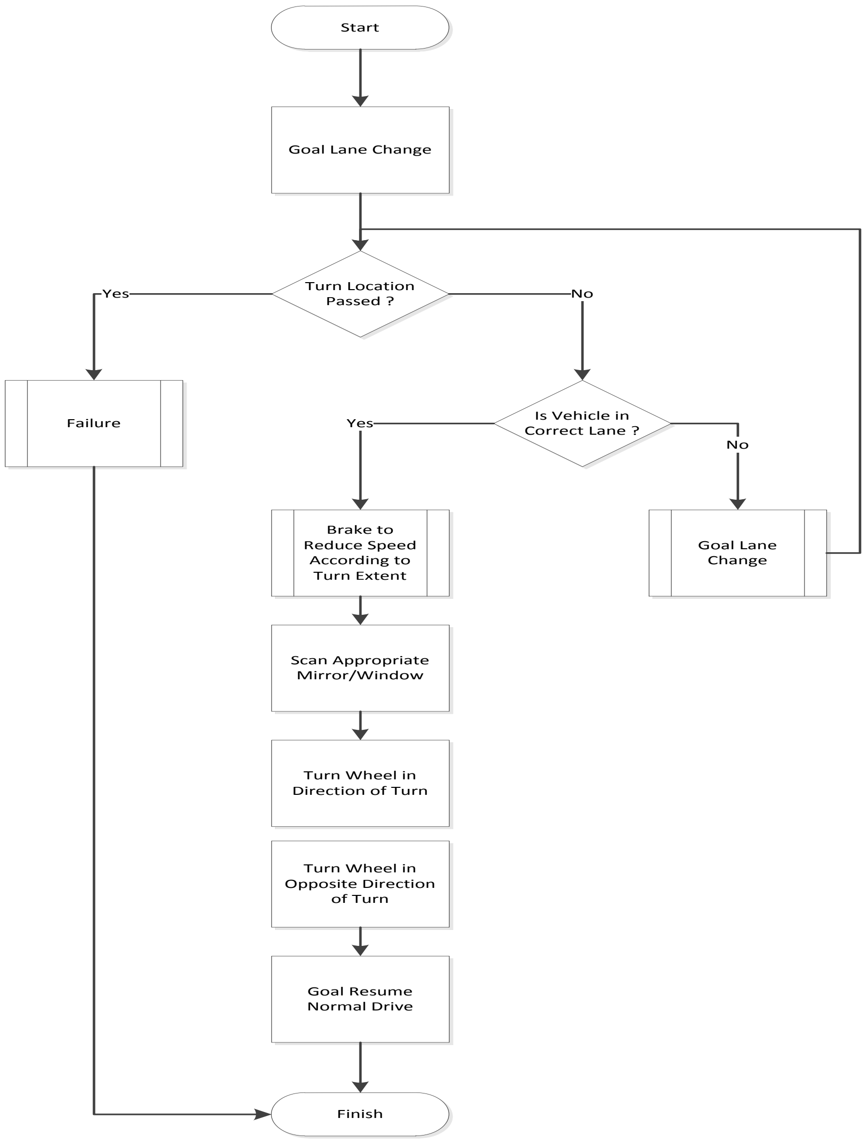
{Wait until traffic light turns green (duration is the time the light will take to turn green from the time that this goal starts executing), then look forward, release brake pedal and hold the wheel steady. This goal represents the driver's intention to start the vehicle moving again after it stops at a light but not to accelerate to normal speed.}



Flowchart of G-start-after-stopping-when-traffic-light-turns-green

G-turn-without-stop (extent: 1-90 degrees, turnlocation: meters, turndirection: left/right, next-road-type: highway/regular/ramp/unchanged, next-road-lanes: integer, next-road-speed-limit: integer, priority: 1/2/3/4, parent: goal instance or null, scheduled-starting-time: GSS-clock, status: scheduled/executing/success/failure)

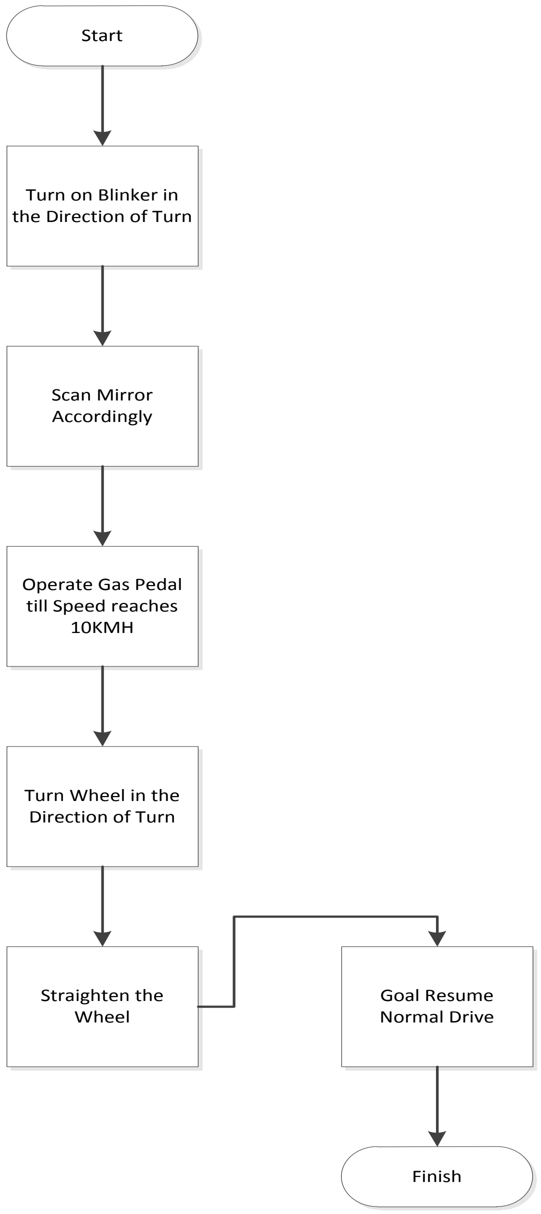
{This Goal is making a turn of up to 90 degrees clockwise (right) or counterclockwise (left) from travel direction. If vehicle has passed the turn by the time driver decides to turn, the goal fails. Otherwise, if not in the correct lane, execute as many lane changes as necessary to get to the correct lane for the turn. If by the time this succeeds the turn location is passed, turn fails. Once in the correct lane too early, start braking. Once the turn is reached, turn on the appropriate blinker, brake to 10 KMH if turn is more than 45 degrees or to 15 KMH if the turn is less than 45 degrees. Scan the appropriate mirror and then out the window and release the brake and execute the turn. When finished, straighten the wheel and resume normal driving.}


Start
Goal Lane Change
Turn Location Passed ?
Yes
No
Failure
Is Vehicle in Correct Lane ?
Yes
No
Brake to Reduce Speed According to Turn Extent
Goal Lane Change
Scan Appropriate Mirror/Window
Turn Wheel in Direction of Turn
Turn Wheel in Opposite Direction of Turn
Goal Resume Normal Drive
Finish

Flowchart of G-turn-without-stop

G-turn-after-stop (extent: 1-90 degrees, location: meters, direction: left/right, next-road-type: highway/regular/ramp/unchanged, next-road-lanes: integer, next-road-speed-limit: integer, priority: 1/2/3/4, parent: goal instance or null, scheduled-starting-time: GSS-clock, status: scheduled/executing/success/failure)

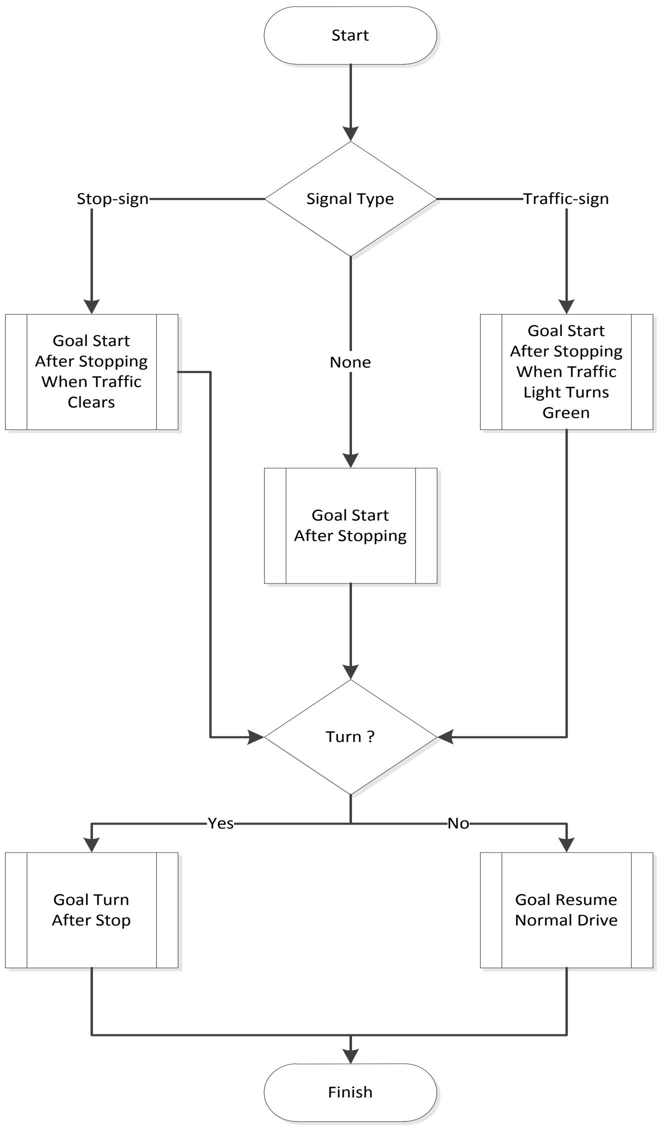
{This goal is about making a turn (up to 90 degrees clockwise/anticlockwise) when the vehicle is already moving at a slow speed, such as when just starting to move after stopping at a signal. If vehicle has passed the turn by the time driver decides to turn, the goal fails. Turn on the directional blinker, scan the appropriate mirror and then out the window, accelerate to 10KMH, execute the turn while watching the road, and straighten out, but not accelerate to the speed limit of the road into which vehicle turned.}


Start
Turn on Blinker in the Direction of Turn
Scan Mirror Accordingly
Operate Gas Pedal till Speed reaches 10KMH
Turn Wheel in the Direction of Turn
Straighten the Wheel
Goal Resume Normal Drive
Finish

Flowchart of G-turn-after-stop

G-drive-from-signal (signal-type: stop-sign/traffic-light/none, duration: milliseconds, turn: Y/N, extent: 1-90, location: meters, direction: left/right, next-road-type: highway/regular/ramp/unchanged, next-road-lanes: integer, next-road-speed-limit: KMH, oncoming-traffic-probability: 0-100, status: scheduled/executing/success/failure, priority: 1/2/3/4, scheduled-starting-time: milliseconds, parent: goal instance or null)

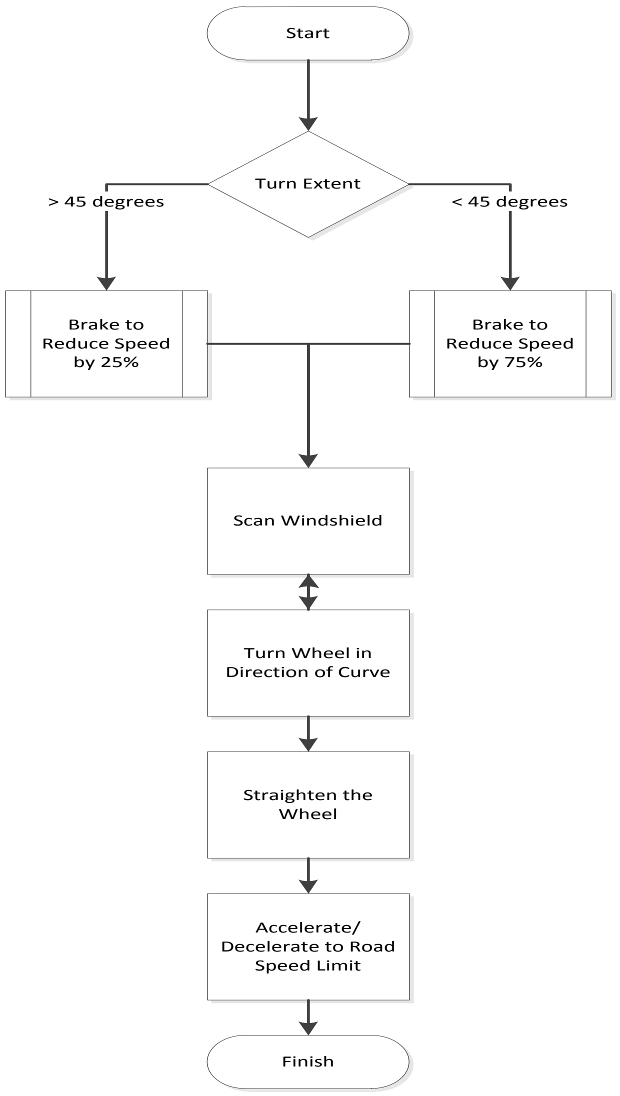
{This goal simulates three possible scenarios after stopping: start going straight after stopping at a traffic light or stop sign, turn and then go straight after stopping at a traffic light or stop sign, or make a turn after stopping for oncoming traffic to clear – signal-type should be stop-sign if stopped at a stop-sign or if stopped for oncoming traffic to clear. This goal triggers the following sub goals: G-start-after-stopping-when-traffic-light-turns-green or G-start-after-stopping-when-traffic-clears depending on signal type followed by G-turn-after-stop if a turn is involved or by G-resume-normal-drive-after-stopping if no turn is involved.}


Start
Signal Type
Stop-sign
Traffic-sign
None
Goal Start After Stopping When Traffic Clears
Goal Start After Stopping When Traffic Light Turns Green
Goal Start After Stopping
Turn ?
Yes
No
Goal Turn After Stop
Goal Resume Normal Drive
Finish

Flowchart of G-drive-from-signal

G-curve (extent: 1-90 degrees, length: meters, direction: left/right, next-road-type: highway/regular/ramp/unchanged, next-road-lanes: integer, next-road-speed-limit: integer, priority: 1/2/3/4, parent: goal instance or null, scheduled-starting-time: GSS-clock, status: scheduled/executing/success/failure)

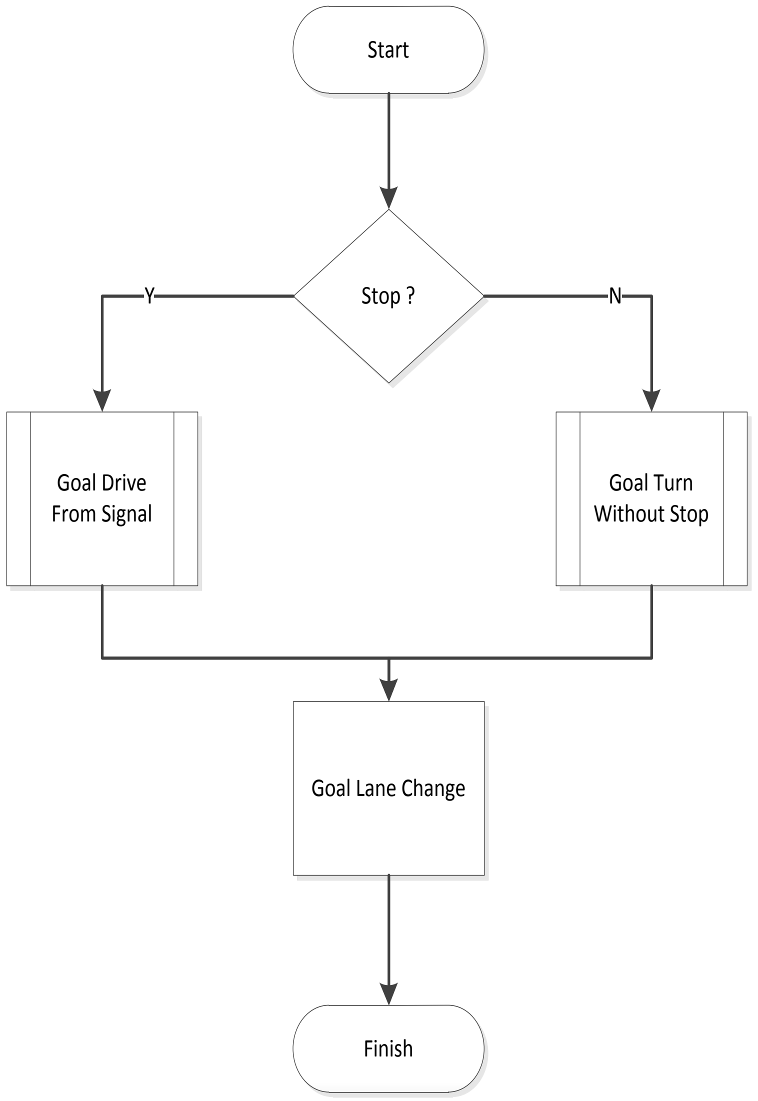
{This Goal is about making a curve of up to 90 degrees. Since curving is part of driving on a road/highway, it is assumed that no lane change is necessary, i.e., either the lane the vehicle is on curves with the road or it turns into another road type. Turn on the appropriate blinker only if the roadway changes, i.e., if simply following the curve of the current road there is no need to turn the blinker on. , Slow down to 0.75 of current speed if turn is less than 45 degrees and to 0.25 of current speed if the turn is higher than 45 degrees. Look out the windshield and simultaneously execute the turn. When finished, straighten the wheel and accelerate or decelerate to speed limit or hold speed steady as appropriate.}


Start
Turn Extent
> 45 degrees
< 45 degrees
Brake to Reduce Speed by 25%
Brake to Reduce Speed by 75%
Scan Windshield
Turn Wheel in Direction of Curve
Straighten the Wheel
Accelerate/ Decelerate to Road Speed Limit
Finish

Flowchart of G-curve

G-enter-highway (oncoming-traffic-probability: 0-100, signal-type: stop-sign/traffic-light, duration: milliseconds, stop: Y/N, turn-extent: 1-90 degrees, distance-to-ramp: meters, ramp-length: meters, direction: left/right, merge-direction: left/right, highway-lanes: integer, highway-speed-limit: integer, priority: 1/2/3/4, parent: goal instance or null, scheduled-starting-time: GSS-clock, status: scheduled/executing/success/failure)

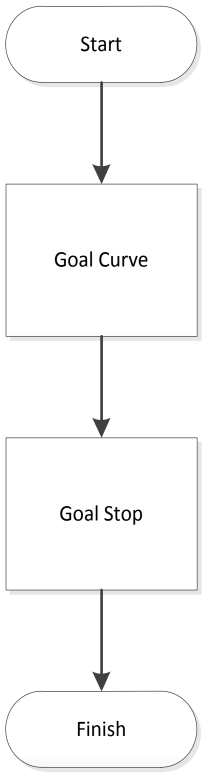
{The vehicle is to either (1) turn left/right without stopping to enter a highway ramp if the stop parameter is N, or it should already be stopped (stop parameter is Y) at a (2) traffic light, (3) stop sign or (4) in a lane in order to make a turn. from which it will enter the ramp. To enter the ramp the simulation will execute G-turn-without-stop in case (1), and G-drive-from-signal for cases (2) - (4). Then execute a merge to a highway lane on the left/right. If merge fails, goal fails and an accident is flagged. Direction is the way to turn from the current roadway to enter the highway ramp; merge-direction is whether the vehicle merges to the left/right onto the highway.}



Flowchart of G-enter-highway

G-exit-highway (turn-extent: 1-90 degrees, ramp-length: meters, direction: left/right, priority: 1/2/3/4, parent: goal instance or null, scheduled-starting-time: GSS-clock, status: scheduled/executing/success/failure)

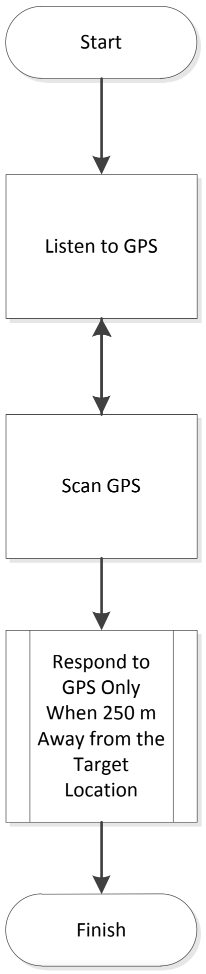
{Execute the appropriate turn goal to enter the ramp and decelerate and come to a stop at a stop sign or traffic signal. The assumption is that events after that, such as starting to move and turning or going straight will be specified separately in the simulation scenario.}



Flowchart of G-exit-highway

G-respond-GPS-command (duration: msec, display-objects: integer, command: traffic-congestion-ahead/traffic-stopped-ahead/lane-closed-ahead/enter-highway/exit-highway/turn-without-stop/stop-before-turn/turn-after-stop, distance: meters, status: scheduled/executing/success/failure, priority: 1/2/3/4, scheduled-starting-time: GSS-clock, parent: goal instance or null)

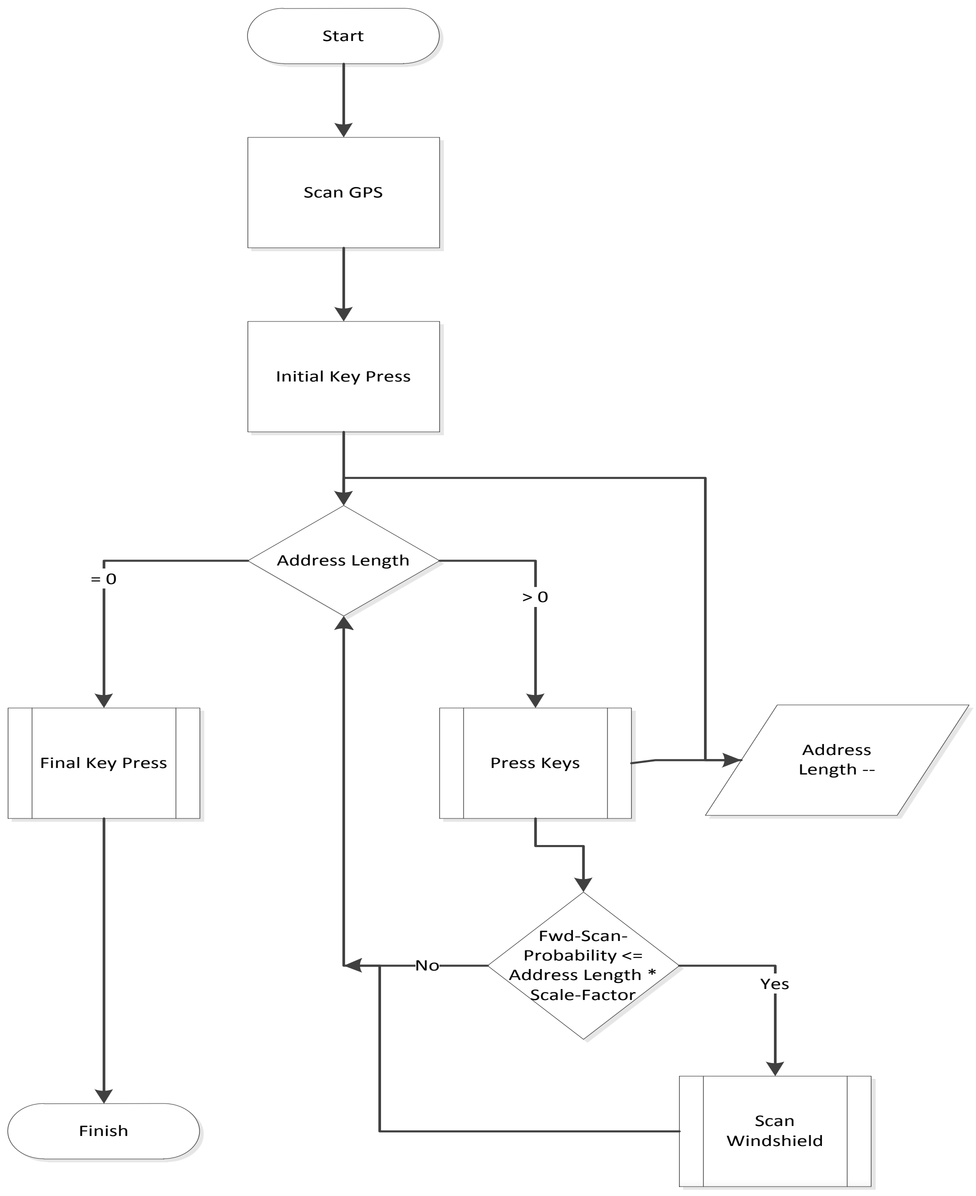
{Listen to GPS spoken command and decide (create goals) on appropriate action. Driver won't start taking action in response to the GPS instruction until the vehicle is a quarter of a kilometer away from the turn or ramp etc., if the GPS command is issued when the vehicle is farther than 250 meters away.}


Start
Listen to GPS
Scan GPS
Respond to GPS Only When 250 m Away from the Target Location
Finish

Flowchart of G-respond-GPS-command

G-enter-address-into-Garmin-Navigation (status: scheduled/executing/success/failure, priority: 1/2/3/4, address-length: integer > 0, scheduled-starting-time: GSS-clock, parent: goal instance or null)

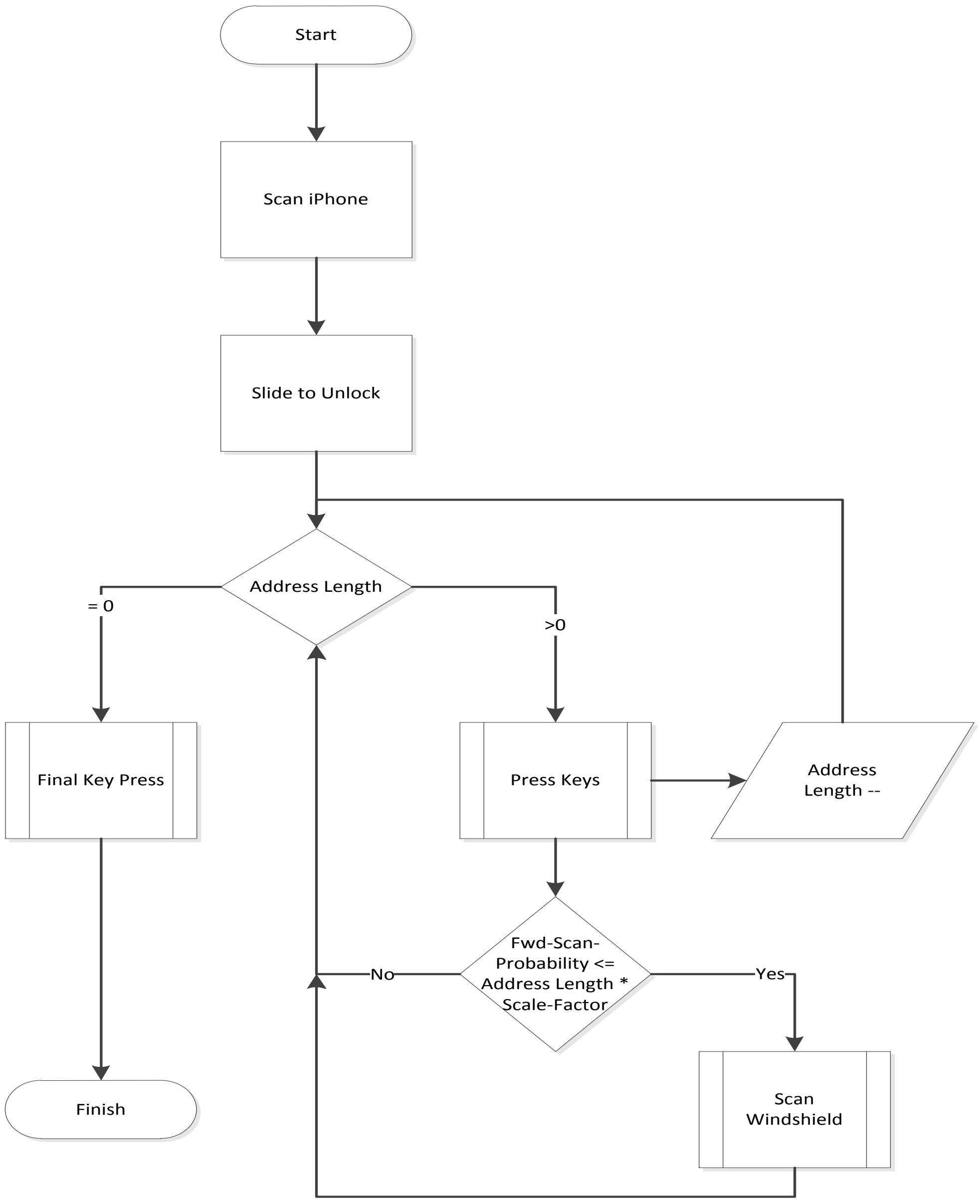
{Look at the navigation system, select the "Where to" and "By Address" sequentially by touching a key on the touch screen while looking at it to active the address-entry display window, enter the destination address one character at a time by touching a key on the touch screen of the navigation while looking at the navigation, after each character entry scan forward with a probability proportional to length of destination address – i.e., the longer the address the more likely is the driver to look forward between typing, finally touch the "search" button while looking at the navigation.}



Flowchart of G-enter-address-into-Garmin-Navigation

G-enter-address-into-iPhone (status: scheduled/executing/success/failure, priority: 1/2/3/4, address-length: integer > 0, scheduled-starting-time: GSS-clock, parent: goal instance or null)

{ Look at the iPhone, slide to unlock the phone, select the "direction" on the touch screen while looking at it to active the address-entry display window, enter the destination address one character at a time by touching a key on the touch screen of the navigation while looking at the navigation, after each character entry scan forward with a probability proportional to length of destination address – i.e., the longer the address the more likely is the driver to look forward between typing, finally touch the "start" button while looking at the navigation.}



Flowchart of G-enter-address-into-iPhone